IMPROVING VISION-BASED ROBOTIC MANIPULATION WITH AFFORDANCE UNDERSTANDING

A Dissertation
Presented to
The Academic Faculty

By

Fu-Jen Chu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2020

IMPROVING VISION-BASED ROBOTIC MANIPULATION WITH AFFORDANCE UNDERSTANDING

Approved by:

Dr. Patricio A. Vela, Advisor
School of Electrical Engineering
Georgia Institute of Technology

Dr. Anthony J. Yezzi
School of Electrical Engineering
Georgia Institute of Technology

Dr. Ghassan AlRegib
School of Electrical Engineering
Georgia Institute of Technology

Dr. Sonia Chernova
School of Interactive Computing
Georgia Institute of Technology

Dr. Eva Dyer
Department of Biomedical Engineering
Georgia Institute of Technology

Date Approved: July 21, 2020

- Goodbye, Hari, my love. Remember always -- all you did for me.

- I did nothing for you.

- You loved me and your love made me -- human.

Isaac Asimov, Forward the Foundation

To my mom, my family,

and the people who have faith in me.

ACKNOWLEDGEMENTS

First and foremost, I would like to deeply thank my advisor, Patricio A. Vela, for his full support of my Ph.D. journey. I really appreciate your guidance over these six years. You are really a patient advisor and always willing to offer me valuable feedbacks. Throughout these years, I have been developing the skill set for conducting research under your guidance. This thesis would not be possible without your insightful suggestions and faith in my abilities. I am especially grateful for your encouragement at some important moments along the way.

I would like to thank Prof. Yezzi for agreeing to be on my thesis committee, and also serving as the chair during my proposal. I would like to thank Prof. AlRegib for taking time out from his busy schedule to be on my committee. I am thankful to my committee member, Prof. Chernova, for providing very helpful feedback on my thesis. I am also thankful to my committee member, Prof. Dyer, for giving me suggestions on my presentation.

It is grateful to meet all kinds of smart and responsible colleagues in Georgia Tech. To Guangcong, Gbolabo and Miguel, thank you all for taking care of me on the first day I joined the lab. To Alex, Luisa, Justin, Yipu and Shiyu, thank you all for willing to share your experience in life and research, and thoughts on all kinds of problems. To Yunzhi and Chao, it is great to have both of you inside and outside the lab. To Ruinian, you are the person I am indebted to. I truly appreciate you for willing to closely collaborate with me. Many parts of my thesis would not be done without your efforts.

Lastly, but most importantly, I am truly grateful to my family. To my grandfather, my grandmother and my father in heaven, you will always be in my heart; thank you for always having faith in me. In particular, I especially appreciate my uncle and my aunt, who treat me as their own son with many kinds of support and invaluable suggestions throughout my academic career. To Yun-Hsuan, thank you for your love and belief in me, even we are thousand miles apart. Finally, I would like thank my mother for her love and support for my educational endeavors. There will never be enough words to describe how blessed I am to have you to be my mother.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of the thesis is to improve robotic manipulation via vision-based affordance understanding, which would advance the application of robotics to industrial use cases, as well as impact the area of assistive robotics.

Research literature related to manipulation primarily focus on the *grasp* affordance due to its essential necessity in robotic manipulation. Recent modern methods for grasp recognition emphasize data-driven machine learning techniques, which improve the generalizability of grasping for novel objects, while falling short of real-time application due to computational cost. Beyond the *grasp* affordance, real-world applications of robotic manipulation involve exploiting more general affordances. Recent studies on detecting object affordances in images address the problem at the pixel-level. Though achieving state-of-the-art performance, per-pixel segmentation approach requires labor-intensive manual annotations. Furthermore, existing affordance datasets contain relatively small quantity of objects, as compared to modern classification datasets. The learned affordances thus only apply to a limited object set. However, a larger variety of objects will be encountered in the real-world. Lastly, given that affordance implies an action opportunity of an object part, the same object part may possess multiple affordances in practical. Recent literature pays attention to single affordance prediction, while ranked affordances of an object part provide flexibility in achieving goal-oriented robotic manipulation tasks.

This thesis focuses on vision-based manipulation for real-time robotic application. A series of methods are proposed to improve the applicability for practical scenarios. Specifically we tackle the problem of identifying viable candidate robotic grasps of objects, and seek for more general affordance map prediction methods with reduced annotation costs. We target generalizing learned affordances to unseen categories and predicting multiple ranked affordance for each object part. We aim to narrow the gap between the vision detection and robotic manipulation by linking action primitives to task execution. To account for various shapes and poses of objects for universal grasp identification, a CNN-based architecture is adopted to learn to grasp. Unlike regression methods,

the identification of grasp configurations in this architecture is broken into a grasp detection process, followed by a more refined grasp orientation classification process, where both processes are embedded within two coupled networks. The final network is applicable in practical scenarios to detect multiple grasps on multiple objects in the view. Beyond grasp affordance, detecting general affordances such as *contain*, *support* and *pound* also benefits robot agent when interacting with physical world. In vision-based affordance detection, finding affordance of an object part is commonly defined as grouping pixels sharing the same functionality. Therefore, the supervised affordance learning process requires pixel-wise annotation ground truth, which is labor-intensive. To reduce the labor-intensive annotation cost, learning from supervised synthetic data with unlabelled real images is considered. To maintain the advantage of jointly optimizing detection and affordance prediction, labelled synthetic data is applied and jointly adapted to unlabelled real images for detection and affordance segmentation. To preserve the advantages of an object-based method while generalizing to unseen categories, a binary classification mode is added for objectness detection and localization. The proposed architecture further adopts KL-divergence to learn the distributions instead of cross entropy for a single label ground truth on each pixel, enabling multiple ranked affordance prediction of one object part. Improvements on affordance prediction is made by proposed branch-wise attention module and attribute-like auxiliary task. A system combining the proposed affordance detector with a pre-trained object detector with the Planning Domain Definition Language (PDDL) illustrates the effectiveness in practical robotic manipulation applications.

Through this research, we study vision-based robotic affordance learning for real-world manipulation scenarios. Methods for identifying graspable areas as well as general affordances are applied to robotic manipulation, improving the training overhead and inference efficiency. For goal oriented tasks, an ideal object with the primary affordance/functionality to achieve the goal might be missing in the view. To compensate, methods for affordance ranking, unseen category generalization and vision architecture improvements are studied, enhancing the flexibility in practical manipulation. In Chapter 2, a multi-object grasping architecture is introduced to enable situations

where no, one, or multiple object(s) are seen, while achieving state-of-the-art on standard benchmarks and physical robot experiments. In Chapter 3, an affordance segmentation architecture is introduced to enable unsupervisedly adapting annotations from synthetic data while achieving comparable performance to supervisedly learned approaches. Considering more realistic scenarios where one object part may support multiple affordances, Chapter 4 extends to multiple affordance with rankings and generalize to unseen categories. In Chapter 5, branch-wise attention module and attribute-like auxiliary task are introduced to improve detection performance on unseen categories. Further integration with object detector and PDDL is introduced to demonstrate applicability in real-world robotic manipulation. Lastly, a case study of the system design is presented in Chapter 6 with proposed components experimented with human subjects for the completeness of this research.

CHAPTER 1

INTRODUCTION

Robotic manipulation is an essential ability for robot agents to physically interact with the world. For humans, visual input serves as the main modality to interpret the world. Vision-based manipulation mainly exploits this perceptual modality for scene understanding in support of robust manipulation. Manipulating objects for the purpose of achieving a desired task requires knowing how objects may be used. Adult humans have a large set of prior experience regarding potential functionalities, or affordances, of objects. This understanding admits successful completion of tasks by identifying which readily available objects contribute to each step in the manipulation sequence. Many applications of robotics benefit from robust manipulations. Resolving it would advance the application to industrial use cases, such as pick-and-place, part assembly, binning, and sorting in assembly lines. Likewise, for assistive robots to support human, more complicated manipulation tasks are required for agents to perceive and interact with their surroundings.

A large fraction of works has focused on exploiting advanced vision algorithms for robotic manipulations. While better visual interpretation to guide manipulation is achieved, few satisfy the need to meet both real-time requirements and practical scenarios. Furthermore, beyond grasp affordance, real-world applications of robotic manipulation involve exploiting more general affordances. However, research in this vein is less prevalent. To improve the applicability of advanced vision algorithm for robotic manipulation, a series of methods are proposed. Specifically we tackle the problem of identifying viable candidate robotic grasps of objects, and seek for more general affordance prediction methods with reduced annotation costs. We target generalizing learned affordances to unseen categories, and predicting multiple ranked affordance for each object part. Furthermore, we illustrate the way to link affordance detection to action primitives for execution with physical robots via a planning module.

In this chapter, we first review the state-of-the-art in robotic grasping using vision. Second,

the literature of general affordance prediction, which serves as the key foundation of the thesis, is presented. Since in this thesis, synthetic data and domain adaptation are covered in proposed approaches, existing works on synthetic data and domain adaptation are reviewed in the third section to complete the literature review. After discussing the gap in vision-based manipulation with affordance, we conclude the section with the outline of this thesis.

1.1 The Gap of Applicability in State-of-the-art Robotic Grasp Detection

Vision-based robotic grasping refers to using a robotic end-effector to securely pick it up without slippage, based on visual perception ability. We study grasping with emphasis on grasp perception, serving as an essential stage for robotic grasping research. We limit the review to vision-based grasp detection, related topics such as haptic grasping and dexterous manipulation (multi-finger with palm) are excluded, since these trends are less related to visual cue understanding.

Research on grasping has evolved significantly over the last two decades. Altogether, the review papers [1, 2, 3, 4, 5] provide a good context for the overall field. Conventional analytical approaches of robotic grasping heavily rely on the knowledge of human experts. Those methods involve manually programming for specific tasks [6], and/or known properties of object geometry, physics models and force analytics as the premise [2]. Though traditional task-specific algorithms developed analytically for grasping are illustrated to be effective, limitations arise in unstructured/changing environments [6]; in certain scenarios, forming a solution with complex analytical approaches becomes arduous and impractical for generalised applicability [7]. Empirical methods or data-driven approaches, on the other hand, rely on collected positive samples. Instead of assuming known object parameters, empirical methods learn from successful object grasps, developing a generalised solution [4] robust to changes in environments. Given the effectiveness of empirically learned grasping and more accessible data, empirical methods have dominated the field of robotic grasping in recent years. Our review of grasping detection hereafter emphasizes learning-based approaches and representation learning.

Empirical grasp methods involve defining a configuration to represent reasonable/successful

grasps from collected samples. Early work in [8] defined a grasping point $g = (x, y, z)$ in a Cartesian coordinate system, and learned a regression model to infer graspable locations. Following the definition in [8], the grasping *region* of an object is defined in [9] to identify a set of graspable points in 3D. The point or region representations suggest suitable object grasps but lack information of predicting a corresponding gripper pose to complete the grasp. To complete a grasp configuration with the associated end-effector, [10] proposed a seven dimensional representation in 3D space, $G = (x, y, z, \alpha, \beta, \gamma, l)$, where (x, y, z) indicates *grasping point*, (α, β, γ) indicates *grasping orientation*, and l is the open width of a parallel plate gripper (or similar in functionality). Simplifying the 7-dimensional representation in [10], a 5-dimensional grasp rectangle of a grasp representation is employed in [11], with the assumption that 2D grasp configurations on images naturally translate to their 3D counterparts. Similar to [10], the 5-dimensional representation described the location, orientation, and opening distance of a parallel plate gripper prior to closing on an object, on 2D plane. The 5-dimensional representation is then adopted by [12] as a bounding box grasp configuration: $g = (x, y, \theta, w, h)$. This 2D orientated rectangle depicts the gripper's location (x, y), orientation θ, and opening distance h. An additional parameter describing the length w completes the bounding box grasp configuration. This 2D oriented rectangle in the image space has shown to provide enough information to guide physical robotic grasping [12] and has been adopted as a well-accepted formulation.

Early work on perception-based learning approaches to grasping goes back to [13], which showed that learning-based methods could generalize to novel objects with low dimensional feature space for identifying grasps. Exploiting the input/output learning properties of machine learning (ML) systems, [9] proposed to learn the image to grasp mapping through the manual designed feature with a probabilistic model. The system was trained using synthetic imagery, then demonstrated successful grasping on real objects; as an end-to-end system, the reconstruction of the object's 3D geometry is not needed to arrive at a grasp hypothesis. Likewise, [14] employed a CNN-like feature space with random forests for grasp identification. In addition to where to grasp, several efforts learn the grasp approach or pre-grasp strategy [15, 16], while some focus on

whether the hypothesized grasp is likely to succeed [17, 18]. Many of these approaches exploited contemporary machine learning algorithms with manually defined feature spaces.

At the turn of this decade, the introduction of low-cost depth cameras enabled models of grasping mapping to encode richer features [19]. Generally, the early methods using depth cameras sought to recover the 3D geometry from point clouds for grasp planning [20], with manually derived feature space used in the learning process. Additionally, the advent of computational frameworks facilitated the construction and training of CNNs. Deep learning approaches [21, 22] were quickly adopted by the computer vision community.

Deep learning avoids the need for engineering feature spaces, with the trade-off that larger datasets are needed. The trade-off is usually mitigated through the use of pre-training on pre-existing computer vision datasets followed by fine-tuning on a smaller, problem-specific dataset. Following a sliding window approach, [11] trained a two stage multi-modal network, with the first stage generating hypotheses for a more accurate second stage. Similarly, [23, 24] first performed an image-wide pre-processing step to identify candidate object regions, followed by application of a CNN classifier for each batch of region. Though the produced result surpassed conventional methods, the speed of inferencing process is impractical for a robotic system.

To avoid sliding windows or image-wide search, end-to-end approaches are trained to output a single grasp configuration from the input data [12, 25, 26]. Regression-based approaches [12] require compensation through image partitioning since the grasp configuration space is non-convex. Following the architecture in [12], [26] experimented on real objects with physical grasping experiments. The two-stage network in [25] adopted residual blocks [27] and first output a learnt feature, which was then used to provide a single grasp output. These approaches may suffer from averaging effects associated to the single-output nature of the mapping. Most deep network approaches mentioned above start with the strong prior that every image contains a single object with a single grasp target (except [23]).

Since multiple grasps may correspond to single object, [28] processed sampled patches of an image with a network followed by a 18-way binary classification of grasp angles. To further

reduce the computational overhead, [29] developed a *Grasp Quality Convolutional Neural Network* to evaluate grasp robustness for each candidate, learning the correlation between object shapes and parallel-jaw grasps. Similar to [29], [30] learns to score each grasp pose predefined on an image plane with simulated depth data. Adapting pixel-wise grasping detection, [31] proposed to regress grasp quality, gripper angle and gripper width for each pixel. With eye-in-hand design, the closed-loop grasping improve the success rate in real-world grasping. A similar eye-in-hand design was also presented in [32] to predict grasp point and angle dynamically adapting for environment changes.

Another line of research is to learn the mapping from vision input to robot motion to achieve grasping. To directly plan grasps, Lu et al. [33] proposed to predict and maximize grasp success by inferring grasp configurations from vision input for grasp planning. Research on empirical grasp planning with reinforcement learning (RL) acquired samples from robots in real experiments [34]. The training time involved several weeks and led to limitation of its scalability. The work [35] collected over 800k data points with up to 14 robotic arms running in parallel for learning visual servoing. The training time involved over 2 months. Recent approaches especially RL-based methods adopt simulation environment for reducing training overhead, but domain shifts exist when applying learned models/policies to real-world [36, 37, 38, 39]. The generalization performance of RL solutions to environmental changes remains unknown.

Though a rich body of works exist in tackling perception-based robotic grasping, we carefully identified the applicability gap of grasp detection to real-world application. Though achieving top performance, recent supervise learned grasp detection algorithms such as [12, 26, 25] hold a strong assumption that a single grasp exists in an image. Guo et al. [40] employed a two-stage process with a feature learning CNN, followed by specific deep network branches (graspable, bounding box, and orientation). The anchor design predicted precise grasp point with bounding box ratio compared to [28, 30] for fixed grasp box or grasp pose with multiple grasp detection. To further focus on features of local patches and avoid computational cost of sliding window, we adopted grasp region proposal network to generate candidate regions for feature extraction. Moreover, to

exploit the classification ability of CNN over regression, we transformed grasp configuration from a regression problem formulated in previous works [12, 25] into a combination of region detection and orientation classification problems with null hypothesis competition. In the proposed work chapter, we describe in detail the performance boost with limited trade-off of computation time, and provide extensive experiments including vision benchmarking and physical grasping test.

1.2 Object Affordances in Images

In ecological psychology, a commonly-accepted definition of *affordance* is defined as "perceived action possibilities" available to an agent on environmental interaction, by J. J. Gibson [41]. The notion of affordance relates cognitive processing to reasoning in artificial systems of robotics. Beyond grasping, robotics research on general affordances studies the applicable interactions between robots and real-world objects/environments [42, 43, 44]. Manipulating objects for the purpose of achieving a desired task requires learning the potential actions to perform on or with an object.

Research literature related to robotic manipulation primarily focuses on the *grasp* affordance [45] over other affordances, due to its essential necessity in manipulation. To extend to general affordance understanding for manipulation, this section covers the review of affordance research. A systematic literature review with comprehensive taxonomy for models of affordance in robotics can be found in [45]. This section covers the literature of image-based affordance detection for robotic manipulation, topics such as haptic affordances, environment affordance, and learning from human demonstration, will not be discussed.

Early works of image-based affordance detection considered affordances as a subset of object attributes [46]. Manually designed features with geometric cues were used in [47] to learn affordance in an unsupervised manner. [48] proposed to leverage local and global shape feature to predict effective contact location. A 3D surface descriptor was utilized in [49] as a representation of affordance with object poses. Similar to [49], a geometrical part-based method was applied to categorize object parts into predefined functionalities.

Recent image-based affordance detection is treated as the problem of pixel-wise labelling of

object parts by functionality. Due to the abstract nature of the concept of affordances, their associations to objects differ from conventional semantic segmentation. For instance, parts with different appearances may hold the same affordance and vice versa. However, this formulation benefits from exploiting advances in semantic segmentation in computer vision community. For pixel-level affordance detection, geometric cues were applied in [47] with manually designed features. A graphical model is proposed in [50, 51] to learn the correlation between affordances and semantically meaningful parts. Adopting the work in [50], [52] proposed to further categorize affordances for manipulation and execution to manipulate a tool for task execution. Following the design of deep CNN for segmentation [53], further improvement was achieved through an encoder-decoder framework with learned feature spaces [54]. A two-stream encoder-decoder framework for multi-model inputs was proposed in [55] with a late-fusion strategy for affordance prediction. Inspired by [56, 57], dense conditional random fields were integrated in [58] with a proposed object-based affordance detector for affordance map prediction, which indicates the affordance prediction benefits from object and bounding box prior. On this front, following the design of [59], [60] proposed jointly optimizing object detection and affordance segmentation for object-based affordance detectors. The end-to-end optimization improves the overall accuracy and achieves state-of-the-art performance.

Due to the cost of labelling the pixel-wise affordance during supervised learning, weakly supervised approaches were introduced in [61, 62] for affordance segmentation with few key point annotations. However, unsupervised learning for pixel-level affordance segmentation for joint detection and segmentation remains open. The studies of the review indicate the benefits of jointly optimizing detection and affordance map segmentation for priors of object class and location. Inspired by [63], to avoid labor-intensive annotation for pixel-wise labelling and detection ground truth, we propose to learn from supervised synthetic data, and jointly adapt to unlabelled real images. This multi-level domain adaptation from synthetic data to real images involves generating corresponding synthetic benchmark and reducing domain shift via Generative Adversarial Networks (GAN) [64]; to complete the review, both synthetic data and domain adaptation will be

covered in the following review sections.

1.3 Synthetic Data and Domain Adaptation

1.3.1 Synthetic data

Synthetic image data has been used for benchmarking the performance of computer vision algorithms. Early in the optical flow area, synthetic data played an important role due to difficulty in real-scene measurements for accurate ground-truth; benchmarking relied heavily on synthetic datasets (MPI-Sintel [65] and Flying [66]). Recently due to the needs for deep reinforcement learning, simulated environments with physical engine [67, 68, 69, 70] defining interaction and state changes were drawing attention [71, 72]. For end-to-end training of grasping in robotic manipulation, simulation environments help to diversify the training data and speed up the training process [36, 37, 38]. The tactic has also been applied to areas where large real-world datasets tend to be harder to acquire, such as autonomous driving [73, 74, 75], among which GTA5 [75] and SYNTHIA [74] are comprehensive and widely utilized for benchmarking. For indoor scenes, SceneNet [76] composed a collection of camera trajectories through random indoor scene configurations under varying conditions. Semantic segmentation involves labor-intensive labelling for ground truth; constructing synthetic dataset has drawn attention to tackle the annotation issue. GTA5 [75] and SYNTHIA [74] provide segmentation mask ground truth and are largely used for benchmarking.

Training with synthetic data reduces annotation costs and shifts the supervision level. At the same time, learning from synthetic data to real data involves domain adaptation due to the visual difference between rendered and captured scenes.

1.3.2 Domain Adaptation

Domain adaptation for image classification has been widely studied in vision community, including kernel learning [77, 78], asymmetric metric learning [79], subspace interpolation [80] and covariance matrix alignment [81, 82]. Recently due to performance gain, CNN based classifiers

are developed to tackle the problem of domain shifts

To reduce the feature distribution mismatch, Domain Adversarial Neural Networks (DANN) [83, 84] align the feature distributions between source and target images. Additional training techniques on classifier and loss designs further reduce the performance mismatch between synthetic and real data [85, 86, 87]. Another line of research adopts generative adversarial networks (GANs) to simulate domain-invariant samples for generating training sets in target domain [88].

Beyond domain adaptation for image classification, domain adversarial learning methods for pixel-level adaptation of image segmentation tasks operate on feature representations [89, 90]. Building on the concept of adversarial learning, class-wise adversarial learning with label transfer was using in [91] for synthetic to real adaptation. In additional to adversarial learning in feature space, multiple levels of output spaces learning was proposed in [92]. [90] adopted cycle-consistent constraint [93] and proposed an architecture with adversarial learning in feature and label spaces.

Compared to image classification and image segmentation, domain adaptation for detection has not been widely explored. For detection with domain adaptation, an adaptive SVM for deformable part model was proposed in [94]. With the progress of deep learning, an R-CNN based feature extractor is utilized in [95] with subspace alignment for domain shift. Recently, a detector for testing across domains was proposed by [63] based on Faster R-CNN [96].

To benefit from the region-based segmentation with object prior, our proposed work follows recent state-of-the-art affordance detection methods [58, 60], for a trainable framework to jointly optimize detection and segmentation. To avoid labor-intensive annotation, we propose to adopt domain adaptation to learn from synthetic data. A simulated dataset was carefully collected with auto-generated labels. Inspired by [63], we propose to learn from supervised synthetic data and jointly adapt to unlabelled real images for detection and affordance segmentation.

Compared to previous works, this work simultaneously localizes candidate objects in the image, predicts object labels, and infers multi-label affordance maps for all object parts. The objective of the framework is to learn from a synthetic UMD dataset with automatically generated ground-truth for the standard UMD benchmark, and to unsupervisedly adapt to real-world manipulation

tasks.

1.4 Outline of the Thesis

The rest of this thesis is organized as follows:

- Chapter 2 introduces a grasp region proposal network for identification of potential grasp regions [97]. The network then partitions the grasp configuration estimation problem into regression over the bounding box parameters, and classification of the orientation angles. The whole network design enables real-time grasp affordance detection for real-world robotic grasping.

- Chapter 3 extends the grasp affordance detection to general affordance prediction for robotic manipulation [98]. The prediction is formulated as segmentation problem. Due to labor-intensive annotation for pixel-level ground truth, an segmentation architecture is introduced to enable adapting annotations from synthetic data unsupervisedly, while achieving comparable performance to supervisedly learned approaches.

- Chapter 4 further extends the general affordance prediction to multiple affordance ranking, on a single object part [99]. Multiple affordance prediction benefits consecutive planning for realistic scenarios where desired tools are absent. Moreover, the designed architecture generalizes learned affordance to unseen categories, improving the applicability of learned model.

- Chapter 5 tackles the performance loss due to generalization of the affordance model on unseen categories [100]. Two modules are introduced to improve the performance: branch-wise attention module and attribute-like auxiliary module. To bridge vision detection and physical robotic manipulation, a system incorporating proposed affordance detector, a pre-trained object detector, and PDDL is introduced. Planning and execution of a sequence of action primitives defined by predicted affordance is enabled.

- Chapter 6 explores an application [101] of previously described components. The whole system design is a case study of a human-in-the-loop system, incorporating the Augmented Reality (AR) glasses, the Tongue-Drive System (TDS) and robotic manipulator with proposed grasp detection. The overall system design is a preliminary study for people with paralysis in upper limb. The chapter details the design and experiments with healthy human subjects.

- Chapter 7 concludes the thesis with a summary of findings and observations.

- Chapter 8 discusses possible future research directions.

CHAPTER 2

MULTI-GRASP AFFORDANCE DETECTION

2.1 Introduction

In this chapter, we describe the work, multi-grasp affordance detection, which aims at predicting graspable locations in practical scenarios for robotic manipulation. To be specific, it considers situations where no, one, or multiple object(s) are seen. Research on vision-based grasp detection exploits the input/output learning properties of machine learning (ML) systems. While deep learning avoids the need for engineering feature spaces, most early attempts on grasp detection adopting CNNs assumes a single object in the view. Some works allowing multiple grasps suffers from slow inferencing time due to sliding window strategy. Diverse and quick detection of robotic grasp candidates for target objects in practical lead to a better grasp path planning, and improve the overall performance of grasp-based manipulation tasks. We aim at adapting region proposal network (RPN) for grasp candidates, and re-formulating regression of bounding box orientation to classification of quantized intervals. In this way the architecture can be end-to-end optimized for grasp features learning and for real-time referencing. By incorporating grasp region proposal network and orientation classification, the end-to-end training of the architecture leads to grasp detection capable of predicting multiple grasps with corresponding confidence scores in real-time physical grasping.

The primary outcome of this work is illustrated in Fig. 2.1. The image shows the main detection result of the proposed networks. The networks take in a RGB-D image as an input and predicts a list of grasp candidates in the view for robotic arm execution. The red lines correspond to parallel plates of the grasping gripper. The white lines indicate the distance between the plates before the grasp is executed. As shown in the image, the network allows simultaneous multi-object, multi-grasp detection, suitable for practical real-world robotic manipulations. Among the existing works,

Figure 2.1: Illustration of the proposed multi-object, multi-grasp detection. The red lines correspond to parallel plates of the grasping gripper. The white lines indicate the distance between the plates before the grasp is executed

Table 2.1: Performance benchmarking on standard Cornell dataset

Approach	**Img-wise**	**Obj-wise**	**Speed**
	accuracy (%)		fps
Jiang et al. [10]	60.5	58.3	0.02
Lenz et al. [11]	73.9	75.6	0.07
Redmon et al. [12]	88.0	87.1	3.31
Wang et al. [23]	81.8	N/A	7.10
Asif et al. [14]	88.2	87.5	–
Kumra et al. [25]	89.2	88.9	16.03
Mahler et al. [102]	93.0	N/A	∼1.25
Guo et al. [40]	93.2	89.1	–
Ours	**95.5**	**91.7**	17.24

our design, as shown in 2.1 achieves the state-of-the-art performance on standard Cornell dataset for both splits. The image-wise split tests the generalizability across instances, while the object-wise split examines the ability to predict grasps on unseen categories. Our network design allows end-to-end training and inferencing. Compared to published works, ours is the first to achieve multiple grasp detection in real-time inference speed ($>$ 16 fps; [25] assumes one and only one grasp in the view). Overall, the proposed networks maintain the best accuracy-speed trade-off for multi-object, multi-grasp detection.

Contributions of this work include:

1) A deep network architecture that predicts multiple grasp candidates in situations when none, single or multiple objects are in the view. Compared to baseline methods, the classification-based

approach demonstrates improved outcomes on the Cornell dataset benchmark, achieving stateof-the-art performance on image-wise and object-wise splits. The code of network is made publicly available [1];

2) A multi-object, multi-grasp dataset is collected and manually annotated with grasp configuration ground-truth as the Cornell dataset. We demonstrate the generalization capabilities of the architecture and its prediction performance on the multi-grasp dataset with respect to false grasp candidates per image versus grasp miss rate. The dataset is made publicly available [2];

3) Experiments with a 7 degree of freedom manipulator and a time-of-flight RGB-D sensor quantify the systems ability to grasp a variety of household objects placed at random locations and orientations. Comparison to published works shows that the approach is effective, achieving a sensible balance for real-time object pick-up with an 89% success rate and less than 0.25 s from image to prediction to plan;

2.2 Background

In chapter 1, related works of the state-of-the-art robotic grasp detection has been reviewed, and the research gap of existing literatures has been carefully identified. Here we briefly summarize the background and research gap, and discuss the most related works.

In this research problem, we are interested in tackling the problem of identifying viable candidate robotic grasps of objects in a RGB-D image. The envisioned gripper is a parallel plate gripper (or similar in functionality). Among most of existing works tackling vision-based robotic grasping, a strong assumption that a single grasp exists in the view is hold, which forms a applicability gap for practical robotic application. Also for works allowing multiple grasp in view, the utilization of sliding window hinders the real-time inferencing during deployment. Maintaining trade-off of computation time and grasp detection accuracy while allowing multiple candidate prediction is essential, as the grasp detection works as the starting point of grasp-based manipulation tasks.

[1] https://github.com/ivalab/grasp_multiObject_multiGrasp
[2] https://github.com/ivalab/graspmultiObject

Deep neural networks have been shown to outperform hand-designed features and reach state-of-the-art performance. The proposed architecture relies on the strengths of deep convolutional neural networks (CNNs) at detection and classification. The identification of grasp configuration for objects is broken down into a grasp detection processes followed by a more refined grasp orientation classification process, both embedded within two coupled networks.

Inspired by [96], we propose to incorporate a *grasp region proposal network* to generate candidate regions for feature extraction. Furthermore, we propose to transform grasp configuration from a regression problem formulated in previous works [12, 25] into a combination of region detection and orientation classification problems (with null hypothesis competition). We utilize ResNet [27], the current state-of-the-art deep convolutional neural network, for feature extraction and grasp prediction. Compared to previous approaches, our method considers more realistic scenarios with multiple objects in a scene. The proposed architecture predicts multiple grasps with corresponding confidence scores, which aids the subsequent planning process and actual grasping.

2.3 Grasp Proposals Network

Given corresponding RGB and depth images of a novel object, the objective is to identify the grasp configurations of potential grasp candidates of an object for the purpose of manipulation. The 5-dimensional *grasp rectangle* is the grasp representation employed [10].

Inspired by *Region Proposal Network* (RPN) [96], the *Grasp Proposal Network* in our architecture (Fig. 2.4) works as RPN and shares a common feature map ($14\times14\times1024$ feature map) of intermediate convolutional layers from ResNet-50 (layer 40). The *Grasp Proposal Network* outputs a $1\times1\times512$ feature which is then fed into two sibling fully connected layers. The two outputs specify both probability of grasp proposal and proposal bounding box for each of r anchors on the shared feature map.

The *Grasp Proposal Network* works as sliding a mini-network over the feature map. At each anchor of the feature map, by default 3 scales and 3 aspect ratios are used for grasp bounding box shape variations, as shown in Fig 2.2c. Hence $r \times 3 \times 3$ predictions would be generated in total.

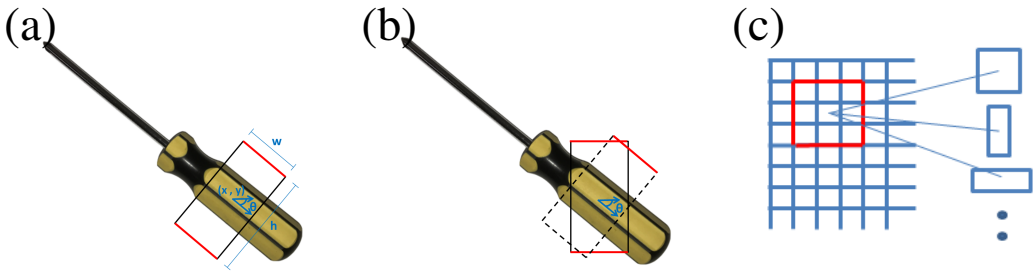

Figure 2.2: (a) The 5D grasp representation. (b) A grasp rectangle is first set to the zero orientation for grasp proposal training. The angle θ is one of the discrete rotation angles. (c) Each element in the feature map is an anchor and corresponds to multiple candidate grasp proposal bounding boxes.

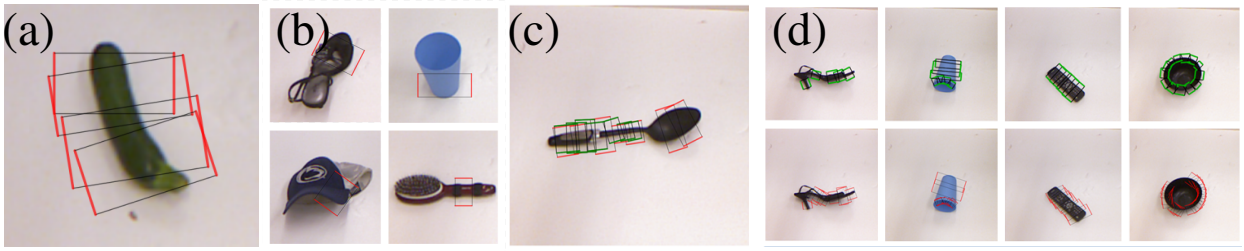

Figure 2.3: Output 5D grasp configuration of system for Cornell dataset inputs: (a) the multiple grasp options output for an object; (b) the top grasp outputs for several objects; (c) output grasps (red) and ground-truth grasps (green) showing that the system may output grasps for which there is no ground truth; (d) multi-grasp output for several objects. The green rectangles are ground truth and the red rectangles represent predicted grasps for each unseen object.

For ground truth, we reset each orientated ground truth bounding box to have vertical height and horizontal width, as shown in Fig. 2.2b. Let t_i denote the 4-dimensional vector specifying the reset (x, y, w, h) of the i-th grasp configuration, and p_i denote the probability of the i-th grasp proposal. For the index set of all proposals $\mathbf{I}$, we define the loss of grasp proposal net (gpn) to be:

$$L_{gpn}(\{(p_i, t_i)_{i=1}^{\mathbf{I}}\}) = \sum_i L_{gp\_cls}(p_i, p_i^*) + \lambda \sum_i p_i^* L_{gp\_reg}(t_i, t_i^*). \tag{2.1}$$

where L_{gp_cls} is the cross entropy loss of grasp proposal classification (gp_cls), L_{gp_reg} is the l_1 regression loss of grasp proposal (gp_reg) with weight λ. We denote $p_i^* = 0$ for no grasp and $p_i^* = 1$ when a grasp is specified. The variable t_i^* is the ground truth grasp coordinate corresponding to p_i^*.

Compared to the widely applied selective search used in R-CNN [103], RPN learns object proposals end-to-end from the input without generating region of interests beforehand. This latter,

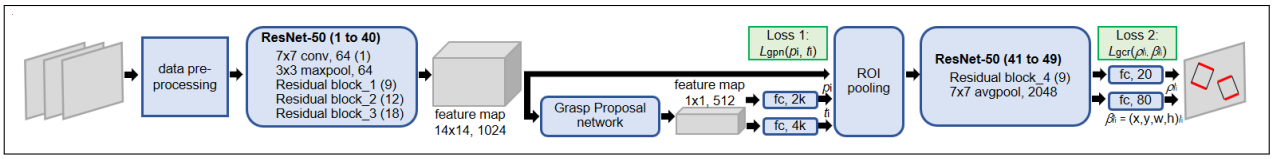

Figure 2.4: Complete structure of our multi-object multi-grasp predictor. The network takes RG-D inputs, and predicts multiple grasps candidates with orientations and rectangle bounding boxes for each object in the view. Blue blocks indicate network layers and gray blocks indicate images and feature maps. Green blocks show the two loss functions. (Best viewed in color)

streamlined approach is more applicable to real-time robotic applications.

2.4 Grasp Orientation as Classification

Many prior approaches such as [25, 12] regress to a single 5-dimensional grasp representation $g = \{x, y, w, h, \theta\}$ for a RGB-D input image. Yet to predict either on $SE(2)$ (planar pose) or on S^1 (orientation) involves predicting coordinates that lies in a non-Euclidean (non-convex) space where regression and its standard L2 loss may not perform well. Rather than performing regression, our multi-grasp localization pipeline quantizes the grasp representation orientation coordinate θ into R equal-length intervals (each interval is represented by its centroid), and formulates the input/ouput mapping as a classification task for grasp orientation. It differs from [40] in that we add a non-grasp collecting orientation class for explicit competition with a null hypothesis. If none of the orientation classifiers outputs a score higher than the non-grasp class, then the grasp proposal is considered incorrect and rejected. In contrast, [40] has a separate grasp confidence score, which may not capture well the orientation-dependent properties of grasps. The value of the non-grasp class is that it is necessary for the downstream multi-object, multi-grasp component of the final algorithm. The total number of classes is $|\mathbf{C}| = R+1$. Denote by $\{(l_i, \theta_i)\}_{i=1}^{\mathbf{I}}$ where the i-th grasp configuration with classification label $l_i \in 1, ..., R$ is associated with the angle θ_i. For the case of no possible orientation (i.e., the region is not graspable), the output label is $l = 0$ and there is no associated orientation. In this work, $R = 19$ is utilized.

2.5 Multi-Grasp Detection

After the region proposal stage of the deep network, the last stage identifies candidate grasp configurations. This last stage classifies the predicted region proposals from previous stage into R regions for grasp configuration parameter θ. At the same time the last stage also refines the proposal bounding box to a non-oriented grasp bounding box (x, y, w, h).

To process the region proposals efficiently, we integrate an *ROI pooling layer* [104] into ResNet-50 so that it may share ResNet's convolutional layers. Sharing the feature map with previous layers avoids re-computation of features within the region of interest. An *ROI pooling layer* stacks all of the features of the identified grasp proposals, which then get fed to two sibling fully connected layers for orientation parameter classification l and bounding box regression (x, y, w, h). The ROI pooling layer receives its input from the intermediate convolutional layer of ResNet-50 (layer 40).

Let ρ_l denote the probability of class l after a softmax layer, and β_l denote the corresponding predicted grasp bounding box. Define the loss function of the grasp configuration prediction (GCR) to be:

$$L_{gcr}(\{(\rho_l, \beta_l)\}_{c=0}^{\mathbf{C}}) = \sum_c L_{gcr\_cls}(\rho_l) + \lambda_2 \sum_c \mathbf{1}_{c\neq 0}(c) L_{gcr\_reg}(\beta_c, \beta_c^*). \tag{2.2}$$

where L_{gcr_cls} is the cross entropy loss of grasp angle classification (gcr_cls), L_{gcr_reg} is the l_1 regression loss of grasp bounding boxes (gcr_reg) with weight λ_2, and β_c^* is the ground truth grasp bounding box.

With the modified ResNet-50 model, end-to-end training for grasp detection and grasp parameter estimation employs the total loss:

$$L_{total} = L_{gpn} + L_{gcr}. \tag{2.3}$$

The streamlined system generates grasp proposals at the ROI layer, stacks all ROIs using the shared feature, and the additional neurons of the two sibling layers output grasp bounding boxes and orientations, or reject the proposal.

2.6 Experimental Results

2.6.1 Dataset and Evaluation Metric

Evaluation of the grasp identification algorithm utilizes the Cornell Dataset [105] for benchmarking against other state-of-the-art algorithms. To demonstrate the multi-object, multi-grasp capabilities, a new dataset is carefully collected and manually annotated [97]. Both datasets consist of color and depth images for multiple modalities. In practice, not all possible grasps are covered by the labelled ground truth, yet the grasp rectangles are comprehensive and representative for diverse examples of good candidates. The scoring criteria takes into account the potential sparsity of the grasp configuration by including an acceptable proximity radius to the ground truth grasp configuration.

Accuracy evaluation of the grasp parameters involves checking for proximity to the ground truth according to established criteria [12]. A candidate grasp configuration is reported correct if both (1) the difference of angle between predicted grasp g_p and ground truth g_t is within 30 °, and (2) the Jaccard index of the predicted grasp g_p and the ground truth g_t is greater than 0.25, are satisfied. Jaccard index is defined as the intersection of g_p and g_t over the union of g_p and g_t.

Typical output of the system is given in Fig. 2.3a, where four grasps are identified. Limiting the output to a single grasp leads to the outputs depicted in Fig. 2.3b. In the multi-grasp case, our system not only predicts universal grasps learned from ground truth, but also contains candidate grasps not contained in the ground truth, Fig. 2.3c.

2.6.2 Single-object Single-grasp

Testing of the proposed architecture on the Cornell Dataset, and comparison with prior works lead to Table 2.2. For this single-object/single-grasp test, the highest output score of all grasp candidates output is chosen as the final output. The proposed architecture outperforms all competitive methods. On image-wise split, our architecture reaches 96.0% accuracy; on object-wise split for unseen objects, 96.1% accuracy is achieved. We also tested our proposed architecture by replacing ResNet-50 with VGG-16 architecture, a smaller deep net with 16 layers. With VGG-16, our model

Table 2.2: Single-Object Single-Grasp

Approach	**Img-wise**	**Obj-wise**	**Speed**
	accuracy (%)		fps
Jiang et al. [10]	60.5	58.3	0.02
Lenz et al. [11]	73.9	75.6	0.07
Redmon et al. [12]	88.0	87.1	3.31
Wang et al. [23]	81.8	N/A	7.10
Asif et al. [14]	88.2	87.5	–
Kumra et al. [25]	89.2	88.9	16.03
Mahler et al. [102]	93.0	N/A	∼1.25
Guo et al. [40]	93.2	89.1	–
VGG-16 (RGB-D)	**95.5**	**91.7**	17.24
Res-50 (RGB)	**94.4**	**95.5**	8.33
Res-50 (RGB-D)	**96.0**	**96.1**	8.33

still outperforms competitive approaches. Yet the deeper ResNet-50 achieve 4.4% more on unseen objects. Furthermore, we experiment on RGB images without depth information with ResNet-50 version and both image-wise and object-wise split perform slightly worse than our proposed approach, indicating the effectiveness of depth. The third column contains the run-time of methods that have reported it, as well as the runtime of the proposed method. Computationally, our architecture detects and localize multiple grasps in 0.120s, which is around 8 fps and is close to usable in real time applications. The VGG-16 architecture doubles the speed with some prediction accuracy loss.

2.6.3 Single-object Multi-grasp

For realistic robotic application, a viable grasp usually depends both on the object and its surroundings. Given that one grasp candidate may be impossible to achieve, there is benefit to provide a rank ordered list of grasp candidates. Our system provides a list of high quality grasp candidates for a subsequent planner to select from. Fig. 2.3d shows samples of the predicted grasps and corresponding ground truths. To evaluate the performance of the multi-grasp detector, we employ the same scoring system as with the single grasp, then generate the miss rate as a function of the number of false positives per image (FPPI) by varying the detection threshold (see Fig. 2.5a for the single-object multi-grasp case). The model achieves 28% and 25% miss rate at 1 FPPI for

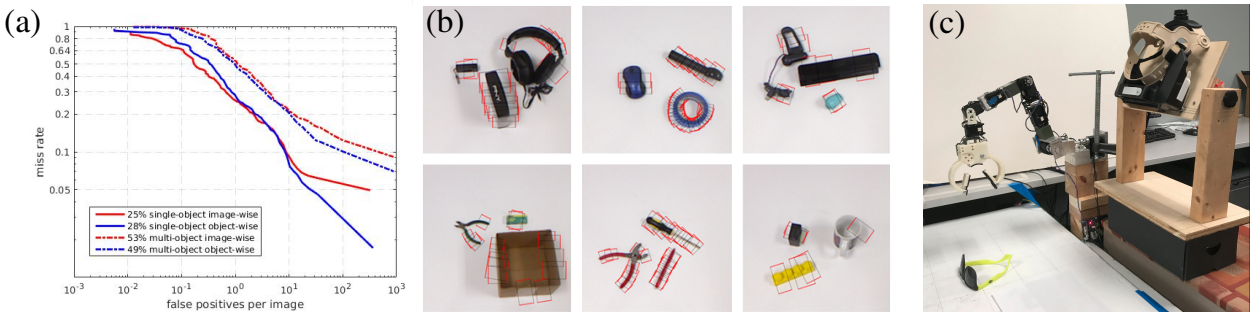

Figure 2.5: Detection results of the system: (a) The ROC curves of our system on single-object multi-grasp scenario and multi-object multi-grasp scenario, respectively. The model was trained on Cornell Dataset and tested on our own multi-object dataset. (b) Detection results of our system on multi-object multi-grasp scenario. The model was trained on Cornell Dataset and tested on our own multi-object dataset. Red rectangle represents the predicted grasp on each unseen object. (c) Experiment setting for physical grasping test. The manipulator is a 7 degree of freedom redundant robotic arm. The vision device is META-1 AR glasses with time-of-flight for RGB-D input.

object-wise split and image-wise split, respectively. A false positive means an incorrect grasp candidate for the object. Thus, accepting that there may be 1 incorrect candidate grasp per image, the system successfully detects 72% (75%) of possible grasps for object-wise (image-wise) split. The model performs slightly better in image-wise split than object-wise split due to unseen objects in the latter.

2.6.4 Multi-object Multi-grasp

For multi-object multi-grasp task, the proposed architecture is applied to test on our Multi-Object dataset. The trained network is the same trained network we've been reporting the results for (trained only on the Cornell dataset with both image-split and object-split variants). Testing involves evaluating against the multi-object dataset, and The testing represents a cross domain application with unseen objects. Fig. 2.5a depicts the plot of miss rate versus FPPI. At 1FPPI, the system achieves 53% and 49% prediction accuracy with image-split model and object-split networks, respectively. Visualizations of predicted grasp candidates are depicted in Fig. 2.5b. The model successfully locates multiple grasp candidates on multiple new objects in the scene with very few false positives, and hence is practical for robotic manipulations.

2.6.5 Physical Grasping

To confirm and test the grasp prediction ability in practice, a physical grasping system is set up for experiments (see Fig. 2.5c). As in [26], performance is given for both the vision sub-system and the subsequently executed grasp movement. The dual scores aid in understanding sources of error for the overall experiment. To evaluate the vision sub-system, each RGB-D input of vision sub-system is saved to disk and annotated with the same protocol as the Cornell and Multi-Object datasets. The evaluation metric uses Jaccard index 0.25 and angle difference 30° thresholds. A set of 10 commonly seen objects was collected from Cornell dataset for the experiment. For each experiment, an object was randomly placed on a reachable surface at different locations and orientations. Each object was tested 10 times. The outcome of physical grasping was marked as pass or fail. Table 2.3 shows the performance of both the vision sub-system and the physical grasping sub-system for two different policies. For the first (Top-1), we used the the grasp candidate with the highest confidence score. For the second, the planner chose the grasp candidate closest to the image-based center of the object from the top-N candidates ($N = 25$ in this experiment).

In real-world physical grasping, grasp candidates close to the image-based centroid of object should be helpful, by creating a more balanced grasp for many objects. The lowest performing objects are those with a less even distribution of mass or shape (screwdriver), meaning that object-specific grasp priors coupled to the multi-grasp output might improve grasping performance.

2.6.6 Indirect Grasping Comparison

Table 2.4 compares our experimental outcomes with state-of-the-art published works with physical grasping. The testing sets for reported experiments may include different object class/instance. Even though the object classes may be the same, the actual objects used could differ. Nevertheless, the comparison should provide some context for grasping performance and computational costs relative to other published approaches. The experiments in [26] had 6 objects in common with ours. On the common subset, [26] reports a 55.0% success rate with a $60s$ execution time, while ours achieves 86.7% with a $15s$ execution time (mostly a consequence of joint rate limits). The

Table 2.3: Physical Grasping Experiment

Approach	**Top-1**		**Nearest center**	
	det	**phy**	**det**	**phy**
banana	10/10	7/10	10/10	8/10
glasses	9/10	8/10	9/10	9/10
ball	10/10	9/10	10/10	9/10
tape	10/10	10/10	10/10	10/10
screwdriver	9/10	7/10	9/10	7/10
stapler	10/10	10/10	10/10	9/10
spoon	9/10	9/10	10/10	10/10
bowl	10/10	10/10	10/10	10/10
scissors	9/10	8/10	9/10	9/10
mouse	9/10	8/10	9/10	8/10
average (%)	**95.0**	**86.0**	**96.0**	**89.0**

Table 2.4: Physical Grasping Comparison

Approach	**Time (s)**		**Settings**		**Success**
	detect	**plan**	**object**	**trial**	
[11]	13.50	–	30	100	84 / **89***
[26]	1.80	–	10	–	62
[28]	–	–	15	150	66
[33]	–	2∼3	10	–	84
[102]	0.80	–	10	50	80
[102]+refits	2.50	–	40	100	**94**
Ours	**0.12**	**0.10**	10	100	**89.0**

* Outcomes are for Baxter / PR2 robots, respectively, with the difference arising from gripper spans.

approach described in [102] reported 80.0% success rate on 10 household objects and 94.0% when using a cross entropy method [35] to sample and re-fit grasp candidates (at the cost of greater grasp detection time). The RL approach taking several weeks achieved 66.0% on seen and unseen objects [28]. Not included in the table are the reported results of [35], due to different experimental conditions. They reported 90% success on grasping objects from a bin with replacement, and 80% without replacement (100 trials using unseen objects). Our approach achieves 89.0% in real-time, with subsequent planning of the redundant manipulator taking 0.1 secs. Overall it exhibits a good balance between accuracy and speed for real world object grasping tasks.

Table 2.5: Ablation Study

Architecture	Cornell Splits		Number of Parameters
	image	object	
(a) RGB	86.4	85.4	24559685
(b) RGB+depth	88.7	86.5	51738757
(c) RGD	88.1	86.0	24559685
(d) RGD+cls*	89.8	89.3	24568919
(e) RGB+cls+gp	94.4	95.5	28184211
(f) RGD+cls+gp	96.0	96.1	28184211

* cls: classification; gp: grasp proposal

2.6.7 Ablation Study

This section reviews a set of experiments, summarized in Table 2.5, examining the contributions of the proposed acrchitecture's components. Firstly, ResNet-50 was used to regress RGB input to 5D grasp configuration output (a). This architecture can be recognized as [12] with a deeper network and without depth information. Then two ResNet-50 networks (b) processed RGB and depth data, respectively, with a small network regressing the concatenated feature for the grasp configuration. This architecture matches [25] and boosts performance. However, the doubled number of parameters results in difficulties when deploying on real-world grasping. To keep the architecture size, one color channel (blue) is replaced with depth information, while the performance is maintained (c). Next, grasp orientation is quantized and an extra branch is trained to classify grasping orientation of an object (d). The last two instances integrate grasp proposals into the ResNet-50 back-bone with added layers, for color (e) and RGD (f) input data. The multi-grasp outputs overcome averaging effects [12] without the need to separate an image into grids. In addition, the RGB-only version of the proposed method is still able to achieve good performance, being slightly worse than including depth information.

2.7 Conclusion

This section presented a novel grasping detection system to predict grasp candidates for novel objects in RGB-D images. Compared to previous works, our architecture is able to predict mul-

tiple candidate grasps instead of single outcome, which shows promise to aid a subsequent grasp planning process. Our regression as classification approach transforms orientation regression to a classification task, which takes advantage of the high classification performance of CNNs for improved grasp detection outcomes. We evaluated our system on the Cornell grasping dataset for comparison with state-of-the-art system using a common performance metric and methodology to show the effectiveness of our design. We also performed experiments on self-collected multi-object dataset for multiobject multi-grasp scenario. Acceptable grasp detection rates are achieved for the case of 1 false grasp per image. Physical grasping experiments show a small performance loss (8.3%) when physically grasping the object based on correct candidate grasps found. This empirically learned grasping detector is robust to detect graspable regions of general objects. However, the outcomes might be improved by fusing the multi-grasp output with object-specific grasp priors. Similar as the two mentioned strategies to select a grasp from the candidate list, task/class oriented strategies may benefit the manipulating tasks. Also, a possible extension is to further train the network on data with positive grasps in the side-view. This may further pushes the grasp detector to be robust in different scenarios.

CHAPTER 3

AFFORDANCE LEARNING VIA SYNTHETIC DATA

3.1 Introduction

In this chapter, we present the work of learning affordance via synthetic data, which tackles the problem of general affordances learning beyond *grasping*. To be specific, the proposed architecture learns multiple common seen affordances from synthetic data to avoid labor-intensive annotation, while adapting to real data for physical manipulation during deployment. Exploiting state-of-the-art convolutional neural network (CNN)-based semantic segmentation architectures, general affordance detection is treated as pixel-wise grouping of object parts based on functionality. literature shows that object category and location information computed by (separate or internal) network components assist affordance prediction. Jointly optimizing both detection and affordance prediction shows positive effect on performance. However it suffers from labor-intensive pixel-level and bounding boxes annotations. Synthetic data has been widely adopted as a simple means to enhance the diversity of data and aid ground truth generation. We aim at design an architecture capable of learning from supervised synthetic data and adapting to unlabelled real images, while jointly optimizing detection and segmentation branches. The problem thus shifts to addressing the performance drop associated to models trained on synthetic data due to domain shift issues when applying to real data. A synthetic version of the UMD dataset is collected for auto-generating annotated, synthetic input data. We show the proposed model achieves performance close to state-of-the-art supervised approaches. Real-world manipulation experiments demonstrate use of the affordance prediction for task execution, which achieves the same performance with supervised approaches.

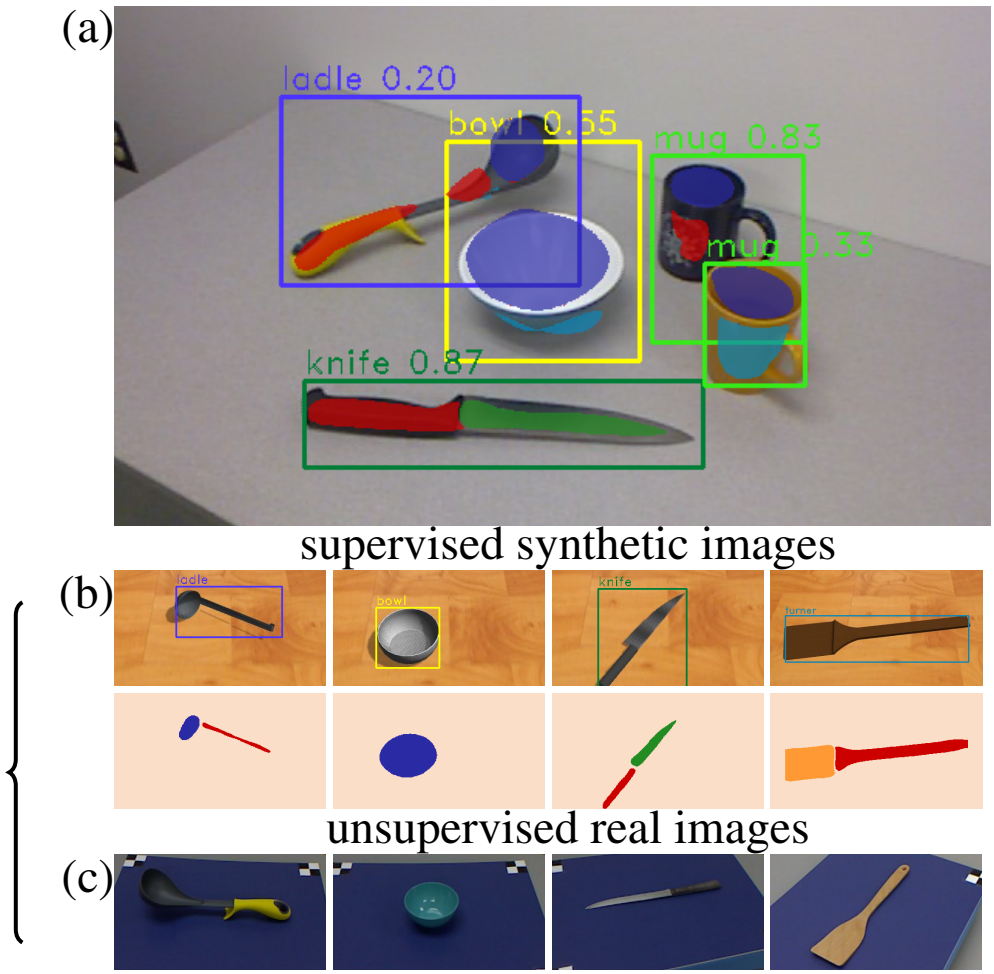

Figure 3.1: (a): Detection of affordance on multiple objects for robotic manipulations using proposed model; (b) and (c): The model is trained on unlabelled real data to avoid annotation cost with supervised synthetic data via proposed domain adaptation components on both detection and affordance segmentation. The color of mask represents afforance. Red: grasp; yellow: scoop; green:cut; dark blue: contain; blue: wrap-grasp; orange: support; purple: pound. Best viewed in color

The primary idea of this work is illustrated in Fig. 3.1. At the top of the figure, the image shows the main detection output of the proposed architecture. The output includes object bounding boxes, category labels, as well as pixel-level affordances. Each segmentation color represents one affordance, or functionality of an object part (best viewed in color). Fig. 3.1c represents the images in the UMD dataset, which is a commonly adopted real-world dataset for affordance learning. Following the same protocol as in UMD dataset, we created a synthetic version of the UMD dataset, as shown in Fig. 3.1b. The proposed architecture is trained on synthetic images

with labelled annotation, and adopted to real data in an unsupervised learning mechanism. Among the existing works, our design achieves the state-of-the-art outperforms an unsupervised baseline on the standard UMD benchmark. Meanwhile, the architecture achieves performance close to state-of-the-art supervised approaches.

Contributions of this work include:

1) A deep network architecture for unsupervised domain adaptation from synthetic images to real-world data. With multi-level domain adaptation components and consistency regularization, it outperforms the unsupervised baseline when annotations are from the source domain. Compared to super- vised approaches, the proposed framework shows reasonable performance on standard UMD dataset benchmark, with a 30.0% performance drop from the oracle. The code of network is made publicly available [1];

2) A synthetic UMD dataset is collected and manually annotated with affordance labels on the model. A toolkit is developed for generating RGB-D images and correspond- ing ground truths including labels, bounding boxes and affordance masks. The proposed framework demonstrates learning from synthetic data for real-world application. The dataset [2] and toolkit [3] are publicly available;

3) Real-world experiments with a 7DoF manipulator and RGB-D camera on manipulations tasks show the proposed approach reaches the same performance as supervised meth- ods. The results demonstrate the ability to learn across domain from simulation to real environment, and the effectiveness of affordance detection on informing manipulation.

3.2 Background

Research on understanding affordances has been widely studied in the robotics community, especially with regards to grasping affordance, which is covered in previous chapters. Beyond grasp-

[1] https://github.com/ivalab/MultiAffordanceNet
[2] https://github.com/ivalab/simData
[3] https://github.com/ivalab/simData_imgSaver

ing, some robotics research on general affordances focuses on the applicable interactions between robots and real-world objects/environments [42, 43, 44].

Exploiting recent advances in computer vision, affordance detection is treated as the problem of pixel-wise labelling of object parts by functionality. For pixel-level affordance detection, manually designed features with geometric cues were used in [47], which was later improved through an encoder-decoder framework with learned feature spaces [54]. On this front, the latest works [58, 60] propose object-based affordance detectors learned through jointly optimizing object detection and affordance segmentation. Affordance segmentation with an object category prior improves the overall accuracy and achieves state-of-the-art performance. However, labelling the pixel- wise affordance with detection annotation during supervised learning is labor-intensive.

Training with synthetic data reduces annotation costs, shifts the supervision level, and helps diversify the training data. In this chapter, we formulate the problem as learning multiple affordance for real-world usage using synthetic data, with the aid of domain adaptation technique. Apart from the affordance learning, we also cover synthetic data and domain adaptation. Synthetic data admits auto-generation of ground truth and readily enables variation of possible changes. In robotics, synthetic data has been widely adopted to generate ranges of inputs needed, especially for the cases where real-data is hard to acquire or to enhance the data diversity. We refer readers to chapter 1 for the literature of synthetic data in robotics. Domain adaptation is a technique to tackle domain shift in spaces. Domain adaptation in vision especially image classification, detection and most importantly segmentation are mainly focused. Chapter 1 covers the review of domain adaptation in above areas. The review concludes that most of adaptation works focus on image classification and segmentation. Domain adaptation on detection are rarely explored, whereas adaptation on jointly detection and instance-level segmentation remains unknown.

Building on [63], we propose to learn from supervised synthetic data and to jointly adapt to unlabelled real images for detection and affordance segmentation. Following recent state-of-the-art affordance detection works [58, 60], a train- able framework is proposed to jointly optimize detection and segmentation. The proposed work simultaneously localizes candidate objects in the

image, predicts object labels, and infers multi-label affordance maps for all object parts. The objective of the framework is to learn from a synthetic UMD dataset with automatically generated ground-truth for the standard UMD benchmark, and to unsupervisedly adapt to real-world manipulation tasks.

3.3 Multi-level Domain Adaptation

To perform multi-label segmentation of objects, it helps to first detect and isolate those objects within the image then to segment them [59]. However, we would like to train such a scheme with additional unlabelled data from a new domain. Using the covariate shift assumption for the joint distribution over the union of the source and target domains leads to domain adaptation mechanisms for the network [63]. Consider the joint distribution over image features I, bounding boxes B, and object class labels C,

$$P(C, B, I) = P(C, B|I)P(I). \tag{3.1}$$

Under the covariate shift assumption, the conditional probability $P(C, B|I)$ is insensitive to source-target domain shift, meaning that cross-domain invariance relies on the marginal distribution $P(I)$ being equal across source-target domain pairings. Hence, the shared image feature map should be domain invariant.

Using a different decomposition of the joint distribution,

$$P(C, B, I) = P(C|B, I)P(B, I), \tag{3.2}$$

then under the covariate shift assumption, source-target domain shift sensitivity is caused by the distribution $P(B, I)$. To alleviate domain shift performance loss, the image region of interest (RoI) detector should be domain invariant.

Finally for the affordance segmentations, we recognize that the affordance labels L have spatial

dependence,

$$P(L(X), B, I) = P(L(X)|B, I)P(B, I) \tag{3.3}$$

where X is the set of pixels in the image domain, indexed by α such that $x_\alpha \in X$ are pixels. Each pixel x_α has the label $L_\alpha = L(x_\alpha) \in \{1, \ldots, l\}$ for l labels. Since not all image regions are relevant, the joint probability is restricted to the regions of interest subset $S \subset X$. Reducing domain shift for pixel-level segmentation requires the RoIs for affordance segmentation to be domain invariant.

We hence consider domain adaptation losses below:

3.3.1 Shared Feature Level

For insensitivity of $P(I)$ from (3.1), the proposed framework employs a fully convolutional domain discriminator $\mathbf{D}$ with adversarial learning, Fig. 3.2(b). The discriminator $\mathbf{D}$ receives the activation feature map of the i-th input image after the base network (the $38 \times 63 \times 512$ feature map in Fig. 3.2) with softmax output $s_i \in \mathbb{R}^{U \times V \times 2}$ ($U = 38$, $V = 63$ in this work). Let z_i denote the domain label of the i-th image, where $z_i = 0$ if the sample is from target domain and $z_i = 1$ for the sample from the source domain. The cross-entropy loss $\mathcal{L}_{\text{shared}}$ on the shared feature map for the two classes, is:

$$\mathcal{L}_{\text{shared}} = -\sum_{i,x} \left[(1 - z_i) \log s_i(x, 0) + z_i \log s_i(x, 1)\right] \tag{3.4}$$

where $s_i(x, 0)$ and $s_i(x, 1)$ denote the domain prediction of activation at the pixel x on the i-th image.

3.3.2 Task Level

Domain discriminators are again used for the task level marginal distributions $P(B, I)$ from (3.2) and (3.3), for which there are two applicable branches in the network after the region pooling layers. For the detection task, the discriminator (Fig. 3.2(c)) operates on the feature vector feeding into two fully connected layers, with $p_{i,j}$ denoting the output of discriminator for the j-th region

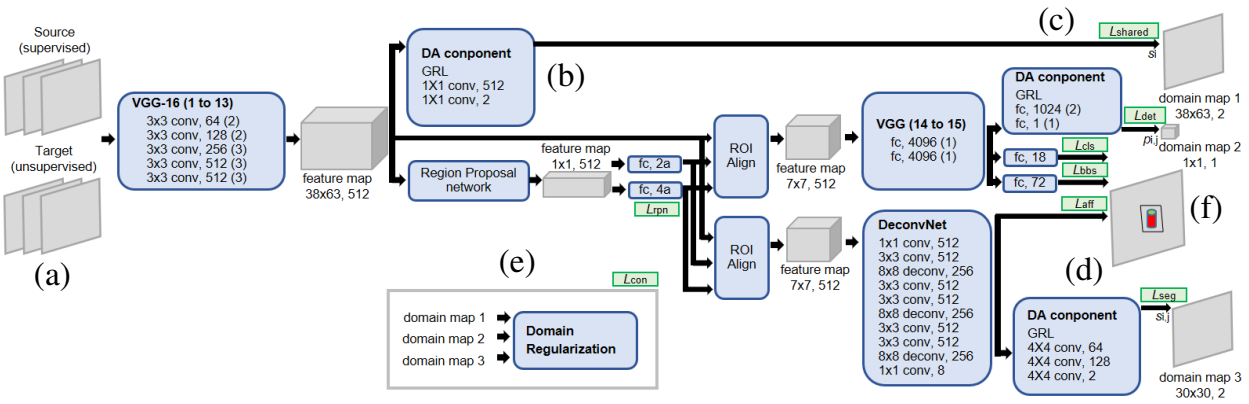

Figure 3.2: Complete structure of the proposed networks for learning object annotation and affordance masks from supervised simulated images to unsupervised real world images. (a): The networks take RG-D images as input either from source or target domain; (b): a domain adaptation component is applied to the output feature map of the whole image; (c) and (d): two domain adaptation components are applied on instance-level feature maps for detection and segmentation, respectively; (e) A regularization term penalizes the inconsistency of the domain predictions from above three domain adaptation components; (f) The final output contains the prediction results of detection and segmentation.

proposal in the i-th input image. The cross-entropy detection branch loss $\mathcal{L}_{\text{det}}$ for domain adaptation is:

$$\mathcal{L}_{\text{det}} = -\sum_{i,j} \left[(1 - z_i) \log p_{i,j} + z_i \log p_{i,j}\right]. \tag{3.5}$$

For the segmentation branch, the discriminator (Fig. 3.2(d)) takes in the RoI-based activation map after the final deconvolutional layer. It operates on the feature map prior to the final output layer for alleviating domain mismatch. Let $s_{i,j}^{(x,0)}$ and $s_{i,j}^{(x,1)}$ denote the domain predictions of activation at x on the j-th region proposal of the i-th image for the source and target domains. The segmentation branch loss $\mathcal{L}_{\text{seg}}$ is:

$$\mathcal{L}_{\text{seg}} = -\sum_{i,j,x} \left[(1 - z_i) \log s_{i,j}(x, 0) + z_i \log s_{i,j}(x, 1)\right]. \tag{3.6}$$

The segmentation and detection branches have different sampling mechanisms for training because the segmentation branch takes in positive samples during training, while the detection branch receives both positive and negative samples.

3.3.3 Joint Adaptation

To improve domain prediction consistency at different levels in the network, we extend the consistency loss from [63] with regularized domain predictions for the base layers and the task branches, Fig. 3.2(e). Let $p_i(x) = s_i(x, 0)$ from (3.4) and $p_{i,j}(x) = s_{i,j}(x, 0)$ from (4.1), denote the predicted confidence of being sampled from the source domain. Define the mean confidence of the j-th proposal on the i-th image for all levels as:

$$p_{\text{shared}}(i, j) = \frac{1}{|X|} \sum_{x} p_i(x) \tag{3.7}$$

$$p_{\text{seg}}(i, j) = \frac{1}{|X|} \sum_{x} p_{i,j}(x) \tag{3.8}$$

$$p_{\text{det}}(i, j) = p_{i,j} \tag{3.9}$$

where $|X|$ is the number of pixels in the region proposal (for p_{shared}, X is the full image). Given the set of discriminator confidences $\mathcal{D} = \{p_{\text{shared}}, p_{\text{seg}}, p_{\text{det}}\}$, their extended consistency loss $\mathcal{L}_{\text{con}}$ is:

$$\mathcal{L}_{\text{con}} = \sum_{\substack{p_{k_1}, p_{k_2} \in \mathcal{D} \\ k_1 \neq k_2}} \sum_{i,j} \|p_{k_1}(i, j) - p_{k_2}(i, j)\|_2^2 . \tag{3.10}$$

3.4 Affordance Mask Loss

Object affordance segmentation differs from semantic segmentation. The latter predicts labels for every pixel regarding the appearance and spatial relation of objects, while the former reflects the functionalities of object parts. Pixels belonging to the same object will have labels associated to the role or functionality of the object part they belong to.

Let $p(x, a)$ denote the predicted affordance mask on a RoI-based feature map. The affordance detection loss $\mathcal{L}_{\text{aff}}$ is defined with multinomial cross entropy loss:

$$\mathcal{L}_{\text{aff}} = - \sum_{x \in RoI} \frac{1}{|RoI|} \sum_{a \in A} Y(x, a) \log(p(x, a)) \tag{3.11}$$

where RoI is the region of interest defined by a bounding box, and $|RoI|$ is its area. Y is the ground truth affordance mask with A channels, where the first channel represents the background, and the remaining channels represent individual affordances. At each pixel $x \in RoI$, only one $a \in A$ has the output 1, and the rest have output 0.

Whereas [59] classifies RoI into foreground and background, the proposed method further classifies foreground into multiple affordances. The pixels considered for the affordance loss lie in a region of interest. With simultaneous adaptation on the detection branch, the pixel-wise affordance segmentation benefits from focusing on a local area and utilizing an object prior.

3.5 Multi-level Adversarial Learning

Compared to the detection branch, the affordance branch has more network layers for fine-grained affordance mask prediction. Adversarial learning of the domain discriminator close to the output layer may be less effective on earlier layers. Auxiliary losses are incorporated in the affordance branch at earlier layers for adaptation enhancement [92, 106], leading to a multi-level adversial learning loss $\mathcal{L}_{\text{seg\_m}}$:

$$\mathcal{L}_{\text{seg\_m}} = \sum_{l} \lambda_{\text{seg}}^{l} \mathcal{L}_{\text{seg}}^{l}, \tag{3.12}$$

where $\mathcal{L}_{\text{seg}}$ is the segmentation loss for domain adaptation defined in (3.6). The multi-level affordance loss $\mathcal{L}_{\text{aff\_m}}$ is:

$$\mathcal{L}_{\text{aff\_m}} = \sum_{l} \lambda_{\text{aff}}^{l} \mathcal{L}_{\text{aff}}^{l}, \tag{3.13}$$

where $\mathcal{L}_{\text{aff}}$ is the affordance detection loss defined in (5.5). The index l ranges over the number of total auxiliary losses applied on the branch in the generalized form and λ_{seg}^{l}, λ_{aff}^{l} denote the loss weightings at each level.

3.6 Experimental Results

3.6.1 UMD Dataset

The UMD dataset [47] contains 28k+ RGB-D images of 105 objects from the kitchen, workshop, and garden. Dataset collection and scanning occured by a Kinect with the object on a rotating table. The UMD dataset covers 17 categories and 7 affordance with pixel-level ground truth affordance label. To obtain ground truth bounding boxes, a segmentation tool is implemented to filter out the background table and obtain object boundaries as rectangle bounding boxes.

3.6.2 Synthetic Dataset

The synthetic dataset has 93 3DWarehouse object models covering the 17 categories and 7 affordances in the UMD dataset, each re-scaled and centered. 37k+ RGB-D images were collected in Gazebo with a simulated Kinect. Model parts were colored by affordance to auto-generate pixel-wise labels and bounding boxes. The annotated models support generating training data (see Fig. 3.3) and placing multiple objects in the view for more challenging scenarios.

3.6.3 Training and Data Preprocessing

The pre-trained VGG-16 [107] initializes the network. To incorporate the geometric information from RGB-D images and reuse the pre-trained weights of VGG-16 [107] on ImageNet [108], the blue channel is substituted with the depth channel as in [12, 97]. Depth channel values are replaced with 0 if missing or NaN, normalized to the range of 0 to 255, with mean value 144. Supervised training first occurs with the synthetic data only and without domain adaptation components. The task branches are trained from scratch and the whole network is trained end-to-end for 2 epochs. The target domain data is then introduced with source data with domain adaptation components incorporated. The whole network is end-to-end trained for another 10 epochs. The learning rate is set to 0.0001, and divided by 10 every 5 epochs. Implementation is in Caffe [109] with cudnn-5.1.10 and cuda-8.0, with training on a single nVidia Titan-X (Maxwell architecture). The training

Table 3.1: Performance on UMD Dataset

	Supervised					Unsupervised	
	HMP [47]	SRF [47]	ED [54]	DeepLab [56]	AffNet [60]	AdSeg [92]	Ours
grasp	0.367	0.314	0.719	0.620	0.731	0.223	0.473
cut	0.415	0.412	0.737	0.600	0.762	0.001	0.599
scoop	0.524	0.481	0.744	0.800	0.793	0.010	0.332
contain	0.810	0.635	0.817	0.900	0.833	0.660	0.830
pound	0.767	0.666	0.794	0.880	0.836	0.001	0.224
support	0.643	0.429	0.780	0.600	0.821	0.041	0.541
w-grasp	0.373	0.285	0.769	0.730	0.814	0.421	0.821
average	0.557	0.460	0.766	0.733	**0.799**	0.194	**0.546**

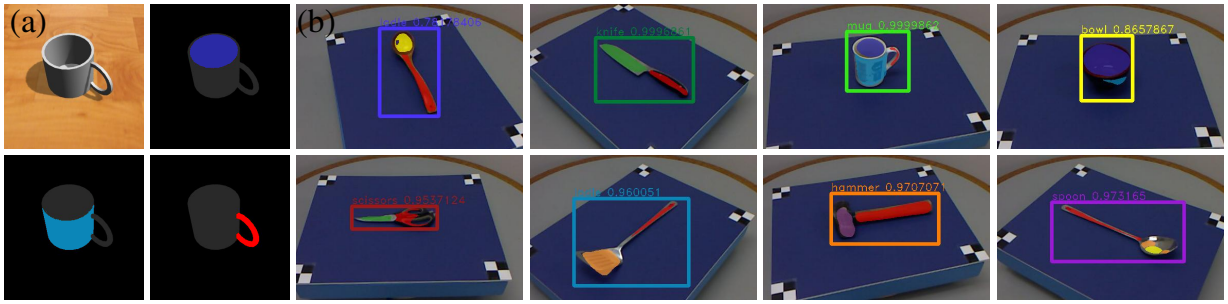

Figure 3.3: (a) Synthetic imagery generated in Gazebo with a 3DWarehouse model; a duplicate with re-painted parts based on their affordance rapidly generates the ground truth masks. (b) Detection results of the system on UMD testing split.

time is around 5 days. The inference time is around 150 ms per image.

3.6.4 Evaluation Metric

The proposed approach outputs a set of affordance probability masks over the input image. Accuracy evaluation involves comparing continuous valued responses against binary valued ground truth labels for each affordance. The F_β^ω metric [110] evaluates the probability masks:

$$F_\beta^\omega = (1+\beta^2)\frac{Pr^\omega \cdot Rc^\omega}{\beta^2 \cdot Pr^\omega + Rc^\omega}. \tag{3.14}$$

where Pr^ω and Rc^ω represent the weighed precision and recall measures, respectively. Pixels close to foreground ground truth are assigned higher weights.

3.6.5 Synthetic to Real-World Domain

Table 3.1 shows the comparison results of our proposed methods with the state-of-the-art *supervised* approaches on UMD dataset and one unsupervised method, AdSegNet[92]. The supervised approaches require the annotated affordance masks, except AffNet [60], which further requires object labels and bounding boxes. AffNet [60] is a generalized version of Mask-RCNN [59] with multiple segmentation labels within each bounding box instance. The GAN-based unsupervised learning approach Cycada [90] was tried. We followed the 2-stage training process to first learn *target domain style* synthetic images for training with real data. The model learns well transformations of backgrounds, but not of objects. The adapted objects are not aligned in shape, pose and location for applying the shared ground truth affordance masks, making the second stage domain adaptation infeasible.

Table 3.1 reports the average F_β^ω score across the 7 affordances. The proposed method, whose results are depicted in Fig. 2.5, outperforms the geometric feature-based approach with random forests (SRF) and is close to hierarchical matching (HMP). To the best of our knowledge, no published work has done affordance segmentation with domain adaptation. Therefore, the unsupervised method AdSegNet is selected due to its state-of-the-art performance on urban scene segmentation with domain adaptation from simulation to real images. The proposed method outperforms AdSegNet, by adapting on segmentation with the aid of detection. Adaptation on detection leverages the local area cue, incorporates the object class prior, and facilitates the performance of pixel-level adaptation within the region proposals. The results indicate the effectiveness of the model structures and the use of domain adaptation on both detection and segmentation.

3.6.6 Real-world Manipulation

To show the ability of the proposed method in practice, a real-world manipulation system is set up for evaluations. As in [60], the performance of grasping and contain are selected to be examined. For physical grasping evaluation, our domain adapted affordance detector recognizes grasping affordance and achieves comparable grasping success rate with supervised learned affNet [58] and

Table 3.2: Physical Manipulation

	[97]	[58]	Ours*	*aff*
knife	10/10	10/10	10/10	grasp
ladle	9/10	9/10	9/10	grasp
mug	N/A	8/10	8/10	contain
bowl	N/A	10/10	10/10	contain
average	**9.5/10**	**9.25/10**	**9.25/10**	

* indicates that annotations are required only for synthetic data during training

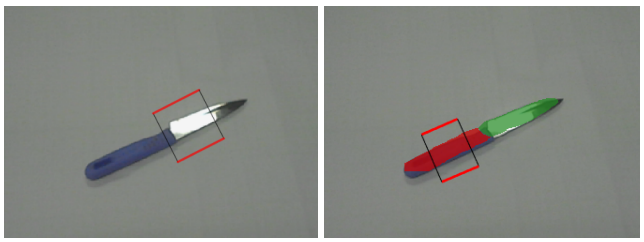

Figure 3.4: A case when state-of-the-art grasp detector may predict a grasp candidate on the blade of the knife (left), while proposed approach alway predict grasp candidate on the handle of knife, due to affordance is taken into account.

state-of-the-art grasp detector deepGrasp [97], as shown in Table 3.2. As shown in Fig. 3.4, it is worthy noting that for the task of grasping knife, although deepGrasp achieves the same success rate as our affordance based grasping, 6 out of 10 predicted grasp candidates were on the blade of knife while ours predicts grasps on the handle of knife in all trials.

Besides grasping tasks in [60], water pouring tasks are performed with a mug using the robotic arm. We showed that with the proposed approach, the robot is able to identify regions within an image with 'contain' affordance for water pouring. We experiment with a robot holding a bottle, and report the success rate of pouring water into the container without spilled out. The results show that our model successfully identify correct region for pouring application in real world scenario, without annotation of real world image data.

The final experiment involving a real robot demonstrates the performance of placing task of the manipulator, where the robot detects a bowl and places a ball into it by detecting "contain" affordance. The proposed approach adapted to the environment for experiment successfully identifies the region for "contain" affordance and achieves the same performance as the supervised learned model.

Table 3.3: Ablation Study

	grasp	cut	scoop	contain	pound	support	w-grasp	average
source	0.037	0.006	0.003	0.346	0.001	0.005	0.339	**0.105**
source+DA	0.326	0.394	0.272	0.761	0.044	0.280	0.783	**0.409**
source+DA+Cons	0.339	0.450	0.273	0.764	0.160	0.328	0.827	**0.449**
source+DA+pairCons	0.447	0.436	0.242	0.843	0.171	0.434	0.821	**0.485**
source+multiDA+pairCons	0.473	0.599	0.332	0.830	0.224	0.541	0.821	**0.546**
oracle	0.709	0.740	0.783	0.828	0.810	0.793	0.795	**0.780**

3.6.7 Ablation Study

Table 3.3 shows the ablation study of the proposed approach. The oracle is equivalent to supervised trained on UMD dataset, indicating the best performance possible for our network. The first column shows the result when trained on the Gazebo synthetic data and tested on UMD directly without domain adaptation. The rest of the columns show the performance of unsupervised learning with different proposed domain adaptation blocks added one-by-one. Although the average score steadily increases as the blocks are incorporated to the model, some techniques slightly decrease the performance of specific affordances. This may be due to cross-affordance coupling of the F_β^ω metric. Given the fact that [92] reports performance drops of 34% and 36% from the supervised oracle to unsupervised DA approaches on two different benchmarks, the 30% performance gap of our proposed approach relative to the supervised oracle is reasonable.

3.6.8 Affordance in the Wild

A more realistic scenario might involve limited accessible annotations of a real-world dataset. To study this case, the proposed method first applies the described unsupervised training method. A finetuning process then randomly samples a subset (50%) of UMD dataset with labels and trains additional 15 epochs. The F_β^ω improves from 0.546 to 0.730. Using 25% of the dataset improves F_β^ω to 0.714.

To further demonstrate the generalizability from the Gazebo syntheic data to other data similar to UMD, the Cornell Dataset [105] is considered. Cornell Dataset is collected for grasp detection and contains objects commonly seen in daily life. Though affordance masks are not available for

Figure 3.5: Detection results on IIT Dataset. The model was trained on synthetic UMD dataset and adapted to real-world images unsupervisedly. The results indicate that more realistic synthetic data are required to adapt to complicated natural images on IIT with changes in occlusion, viewpoints and tool categories.

quantitative evaluation with F_β^ω metric, qualitative results are presented in Fig. 4.5 for readers reference.

Failures of the proposed method are shown in Fig. 3.7. The proposed method performs poorly on a hammer's head (see upper row of Fig. 3.7). We hypothesize the cause as being the various hammer head shapes in the dataset. Furthermore, the proposed approach relies on the detection performance. Affordace detection fails once the wrong bounding box is predicted. Incorrect affordance labels are assigned when the wrong object class is inferred. Balancing the adaptation performance of the detection/segmentation branches requires further investigation.

Additional failures can be exposed by exploring more complex and varied image data, such as that in the IIT-AFF dataset[54]. It contains 8835 images with 10 object categories and 9 affordances. IIT-AFF includes more realistic images with various viewpoints, occlusion and human interactions. Some do not apply to the robotic implementations envisioned. Fig. 3.5 shows some detections results. The average F_β^ω score on the 6 common affordances is 0.021. More effective data generation methods such as domain randomization [111], photorealistic data, and more faithful use cases are required for adapting to these complex settings.

3.7 Conclusion

We presented a novel framework to predict the affordance of objects as semantic segmentation for real world robotic manipulation. Compared to previous approaches, the proposed framework

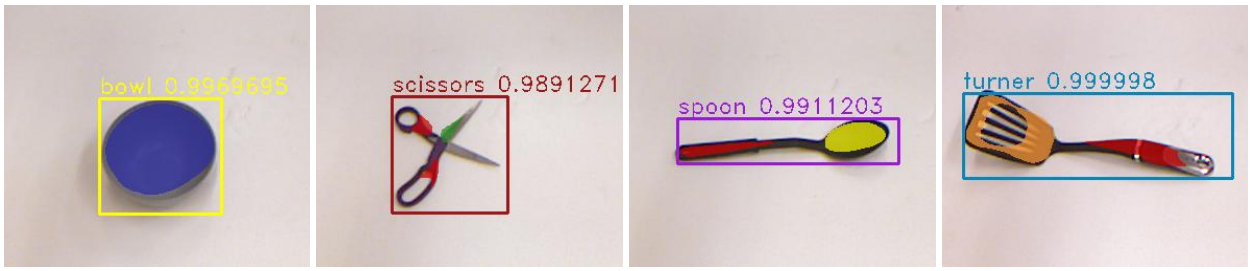

Figure 3.6: Detection results on Cornell Grasping Dataset. The model was trained on synthetic UMD dataset and adapted to real-world images unsupervisedly. The color of mask represents affordance. Red: grasp; yellow: scoop; green:cut; dark blue: contain; blue: wrap-grasp; orange: support; purple: pound. Best viewed in color.

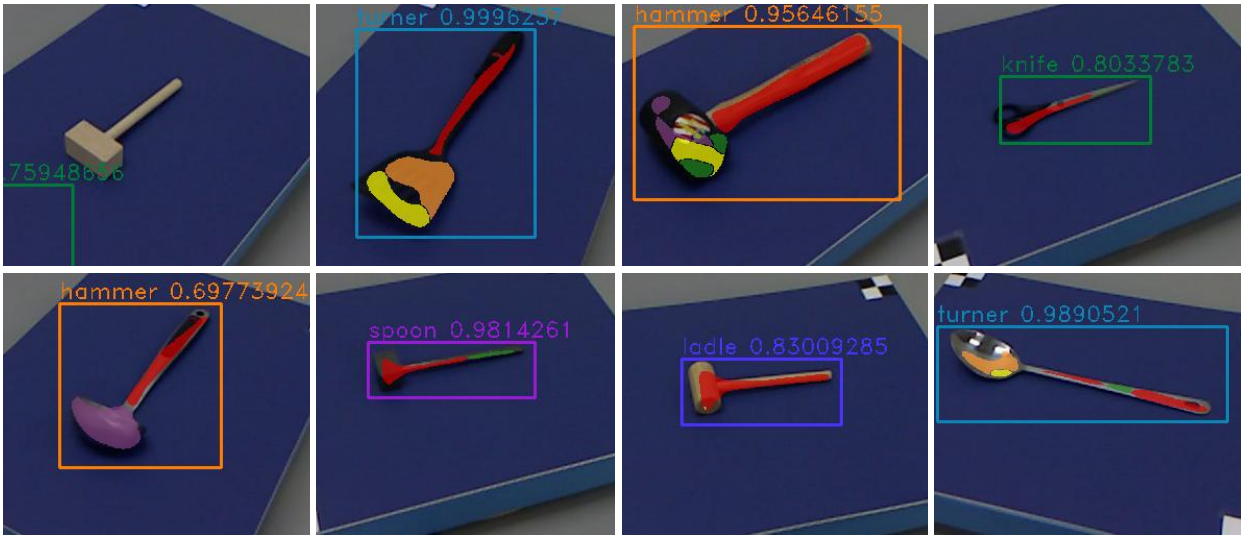

Figure 3.7: Failure cases on UMD dataset. From left to right in upper row: mallet not detected, turner affordances wrongly predicted, hammer affordances wrongly predicted, scissors classified to knife; from left to right in bottom row: ladle classified to hammer, tenderizer classified to spoon, mallet classified to ladle, spoon classified to turner.

is trained in an unsupervised manner without annotation cost. Our framework tackles domain shifts with domain adaptation components in region proposal level and task level, as well as additional proposed regularization to ensure consistency of adaptation in multiple levels. We show the framework is able to adapt annotations from simulated data to real-world images. We evaluate our framework on UMD dataset for comparisons with state-of-the-art supervised methods and unsupervised approach to show the effectiveness. Physical manipulation experiments show the proposed unsupervised framework achieves the comparable performance as supervised learned framework and state-of-the-art grasp detector. A limitation of this work is the performance loss

compared to the oracle method. Although we showed the 30% performance drop is reasonable in the domain adaptation field. Further improvements could be achieved by considering domain randomization to narrow the performance gap. For complicated tasks, more accurate affordance segmentation will be required to identify proper regions for interaction.

CHAPTER 4

LEARNING TOWARD MULTI-AFFORDANCE AND RANKING

4.1 Introduction

In this chapter, we extend the object affordance prediction developed in previous chapters, and propose a general affordances detection and ranking framework for more realistic scenario in robotic applications. To be specific, this extended framework generalizes to unseen categories, and predicts a ranking of detected affordances on an object part.

The object-based affordance detection framework, as introduced in previous chapters, benefits the pixel-level affordance prediction from localizing objects in images and identifying object classes. This advantage, however, limits the learned model from applying to unknown categories, due to an object is required to be in the list of training set, while a larger variety of objects will be encountered in real-world robotic applications. Another limitation of the model is the single affordance prediction of each object part. Robotic affordance implies possible functionalities, or action opportunities of an object part. Thus multiple affordances may exist on the same object part, to serve different manipulation purposes. As an example, the primary affordance for a bowl serves to *contain*, but it can be used to *scoop* when needed. We aim at decoupling the object class from the localized affordance labels of object parts. In the mean time, we propose to learn the distribution of affordances on an object part, instead of predicting the top affordance. Generalizing to novel categories and learning non-primary affordances provide flexibility to achieve manipulation goals. Predicting a ranked affordance list of object parts will benefit the planning stage when the ideal tool is not available.

The primary idea of this work is illustrated in Fig. 4.1. We show the framework of proposed affordance detection with PDDL for real-world manipulations. In practical, for a goal-oriented task, ideal candidates to achieve a goal may not exist. Thus identifying multiple affordances on

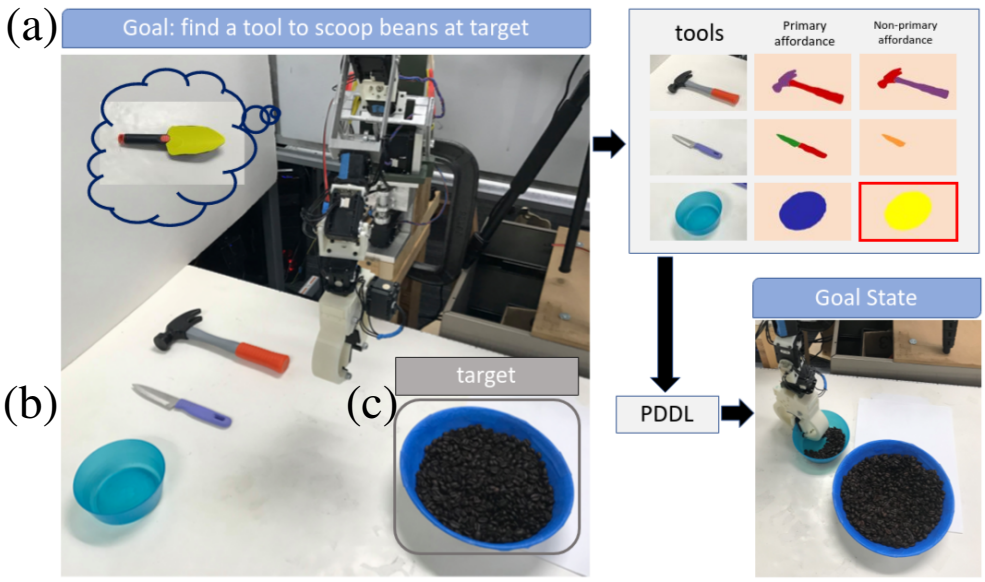

Figure 4.1: Illustration of the proposed multi-affordance detection framework with PDDL [112] for goal-directed robotic manipulation. **(a)** The goal is to scoop coffee beans from the large blue bowl using a tool with the *scoop* affordance. In practice, ideal candidates such as a trowel (with *scoop* as primary affordance) may not be present; **(b)** The proposed affordance detector predicts multiple affordances on a singe object part, improving the chance of finding tools with the required functionality (red boundary indicates affordance found); **(c)** Detected objects and affordances define an initial state and object-action relations for PDDL. Given the goal state, a sequence of action primitives is planned and executed to complete the task.

a singe object part benefits the chance of finding substitutes with the required functionality. In Fig. 4.1b, the secondary affordance of a bowl is *scoop*, which is able to substitute ideal tools such as trowel to complete the tasks of scooping coffee beans. In Fig. 4.1c, affordances implies action opportunities on an object part. Thus detected objects and affordances form initial state and object-action relations. With PDDL and a given goal state, a sequence of action primitives is planned and executed to complete the task.

Contributions of this work include:

1) A deep network architecture to perform category-agnostic affordance segmentation for manipulation. The learned affor- dances generalize to unseen object categories. Compared to previous methods, the proposed framework achieves state- of-the-art performance on the UMD benchmark;

2) The proposed KL-divergence loss enables multiple, ranked affordance predictions for the same object part. The performance of prediction of ranked affordance outperforms previous approaches

on the UMD dataset;

3) Real-world manipulation experiments include utilizing PDDL-generated action primitives from predicted affor- dances, demonstrating the effectiveness of the proposed architecture for informing manipulation. The results demon- strate that the ability to predict multiple affordances across unseen objects benefits real-world robotic manipulation.

4.2 Background

Following the robotic affordance background described in chapter 3, we continue to discuss related literature in this section. To recap, building on advances in computer vision, affordance detection has been casted as the problem of assigning labels to object parts according to their functionality. Hand-engineering features based on object geometry are utilized in [47] to learn pixel-wise affordance of objects in images, followed by [54] with learned features using Convolutional Neural Networks (CNN) from RGB-D data. In [58], the authors propose a two-stage detector to localize and crop objects from images and perform segmentation. Inspired by Mask-RCNN [59], [60] improved object-based methods by jointly optimizing detection and affordance segmentation task branches.

With a boost from a strong object category and localization prior, object-based methods achieve the state-of-the-art performance. Nevertheless, the object category prior also limits the generality to apply or transfer learned affordances across categories. Affordance is defined as potential action choices with objects given their capabilities and the environment [41]. The problem becomes more difficult when multiple affordance labels may be assigned to a pixel. Given the nature of affordances, it is likely that a given object part may serve different purposes under different conditions, especially in the context of manipulation in the real-world.

To support multiple labels per pixel, hierarchical matching pursuit (HMP) was proposed in [113] and adopted in [47]. Superpixel-based HMP and structure random field (SRF) were used in [47] to learn local shape and geometry primitives for generalized affordance prediction and ranked

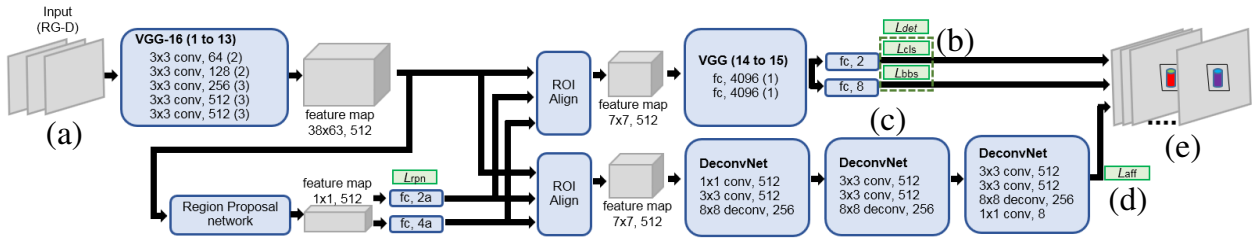

Figure 4.2: Network structure of the proposed multi-affordance detector. The network predicts ranked affordances of object parts for each object in the view. Blue blocks indicate network layers and gray blocks indicate images and feature maps. **(a)** RG-D images are input of the network; **(b)** Proposals with *objectness* are identified with binary classification for category-agnostic affordance segmentation; **(c)** Three concatenated deconvolutional layers lead to a fine-grained ($224 \times 224 \times 8$) affordance map; **(d)** Each RoI-based feature map is applied with KL-divergence loss across affordance to learn distributions for affordance ranking; **(e)** The final output includes bounding boxes and multiple layers indicating confidences for multiple affordances on a single pixel.

affordance labelling of object parts. Multiple affordance labelling aids manipulation planning since understanding object-action relations aids the specification of planning action primitives to achieve a given goal state. Scene estimation methods have been proposed in [114, 115] with RGBD sensor for building axiomatic scene graphs and abstraction for planning. [116] proposed to retrieve inter-object relations for goal-directed manipulation [117, 118]; initial/goal scene graphs based on robot observations are generated by PDDL [112] for planning an action sequence via STRIPS [118].

Extending [60], this work hypothesizes that non-primary affordances should be object-agnostic attributes. As a consequence, affordance transferring to unseen object categories can be achieved. The proposed architecture adopts an object-based pipeline to retain the advantages over image-based methods for affordance segmentation but performs a separate pixel-wise assignment independent of object class for generality. Furthermore, the proposed architecture uses KL-divergence for multi-affordance prediction on each pixel. The learned distribution admits ranking candidate affordances on object parts. Lastly, the affordance detections are integrated into a PDDL planning framework to demonstrate the usage of a real robot with perception of object-action relations to complete manipulation tasks through non-primary affordances.

4.3 Category-Agnostic Affordance Segmentation

To achieve affordance detection and ranking on novel objects, we propose a multi-affordance detection framework, incorporated with PDDL for goal-oriented robotic manipulation. In chapter 4.3, 4.4 and 4.5, we first describe the deep network design, depicted in Fig. 4.2, Chapter 4.6 concludes with a review of PDDL-based planning, and the method to sequence action primitives from detected affordances for robot manipulations.

Isolating objects within an image through an object detection branch benefits object segmentation tasks [59]. The object detection stage provides object prior and helps multi-label segmentation for affordance map prediction [58, 60]. However, categorical annotation can be time-consuming and labor-intensive. We integrate a detection branch for localizing objects while learning a category-agnostic segmentation branch by reducing detection to binary classification.

The binary classifier identifies **objectness** (foreground vs. background), hence the number of classes $|\mathbf{C}|$ is 2. Let ρ denote the probability of being foreground after a softmax layer, and β denote the corresponding objectness bounding box. Define the loss function of detection (det) to be:

$$L_{det}(\{(\rho,\beta)\}_{c=0}^{\mathbf{C}-1}) = \sum_{c} L_{cls}(\rho) + \lambda \sum_{c} \mathbf{1}_{c\neq 0}(c) L_{bbs}(\beta_c, \beta_c^*). \tag{4.1}$$

where L_{cls} denotes the classification (cls) loss using cross entropy, and L_{bbs} denotes the bounding box (bbs) loss using l_1 regression with weight λ. β_c^* denotes the ground truth bounding box annotation. With binary classification, affordance map prediction benefits from the localizing objects within an image while being independent of object class.

4.4 Ranking Loss with KL-divergence

To detect object affordances within images at the pixel level, pixels sharing the same functionality are grouped and segmented in a predicted affordance map. In [59], the segmentation map of an instance is a binary prediction (foreground and background). To predict affordance for each of the instance part, [58] proposed multi-layer output for multi-label prediction. In addition to

primary/part-specific affordances corresponding an object, each part of an object may admit multiple affordances. For instance, the primary affordance of a bowl is *contain*, though it can be also used to *scoop* absent a better object.

To predict ranked affordance on each pixel of segmentation, we propose to use the KL-divergence to learn the corresponding distribution for ranked outputs. Let P denotes the real distribution and Q denotes the predicted distribution. The Kullback-Leibler (KL) divergence is:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log(\frac{P(x)}{Q(x)}) \tag{4.2}$$

In practice, minimizing equation (4.2) is equivalent to minimizing the KL divergence loss L_{kl}:

$$L_{kl} = -\sum_{x \in X} P(x) \log(Q(x)) \tag{4.3}$$

Specifically, let x denote the x-th pixel of an image and $p(x, a)$ denote the predicted confidence of affordance a on the mask. For an RoI-based feature map, the affordance branch loss, modified from equation (4.3), is defined as:

$$\mathcal{L}_{\text{aff}} = -\sum_{x \in RoI} \frac{1}{|RoI|} \sum_{a \in A} p^*(x, a) \log(p(x, a)), \tag{4.4}$$

where RoI (region of interest) is determined by a bounding box and $|RoI|$ defines total area. p^* is the ground truth distribution over A individual affordances, including the background.

When a pixel $x \in RoI$ has only a primary affordance $a \in A$, the output is 1. If a second affordance is defined, a_{21} and a_{22} are set for primary and the second affordance, respectively. Lastly, a_{31}, a_{32} and a_{33} are set for the three affordances case. In this work, a_{21} and a_{22} probabilities are set to 0.6 and 0.4, while a_{31} a_{32} and a_{33} probabilities are set to 0.6, 0.3, and 0.1. These selected settings do not reflect an actual distribution, but serve to induce a priotization and ranking. By minimizing KL-divergence loss, the proposed architecture predicts pixel-wise affordance rankings.

4.5 Fine-grained Deconvolution Mask

The mask head added in Mask-RCNN [59] utilized masks with small spatial resolution for instance segmentation in binary classification: 14×14 for ResNet [27] backbone and 28×28 for FPN [119]. For multi-class affordance prediction, a high resolution mask improves the prediction results in each of the object part [58].

Inspired by [58], we adopt deconvolutional layers to upsample the feature map of the backbone architecture repeatedly for proposed multi-class, multi-output affordance prediction with a high resolution output map. Instead of predicting a 14×14 low resolution binary map in [59], a 244×244 affordance map is predicted with consecutive deconvolution operations. To be specific, three deconvolutional layers are utilized. The first two deconvolutional layers consist of stride $s = 4$ with kernel size $k = 8$. The third deconvolutional layer contains stride $s = 2$ with kernel size $k = 4$.

The proposed networks modifies Faster-RCNN [96] with a VGG16 [107] backbone. The framework inherits the original loss for the region proposal network (RPN) from Faster-RCNN. The original classification loss and bounding box loss are modified in equation (4.1) as $\mathcal{L}_{\text{det}}$. The additional loss $\mathcal{L}_{\text{aff}}$ is adopted for multiple affordance prediction. Let $\mathcal{L}_{\text{rpn}}$ denote the original loss RPN. The final loss for the overall pipeline is:

$$\mathcal{L}_{\text{aff}} = \mathcal{L}_{\text{det}} + \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{aff}} \tag{4.5}$$

4.6 Planning with PDDL

During manipulation robots may need to complete a task by interacting with objects in the environment to achieve the goal state. The initial state of the local envionrment can be estimated by predictions of objects and affordances in the robot's view. To complete desired manipulation tasks, we aim to generate action primitives as a plan for achieving manipulation goals from the observations.

Planning Domain Definition Languange (PDDL) has been widely used as a standard encoding language for planning tasks. By pre-defining the **object** and corresponding affordance as **predicates**, a **domain** can be established to describe the actions and the corresponding results. The proposed affordance detection architecture contributes to a **problem** by describing the initial state of the world for planning operations. The PDDL takes the **domain** and **problem** then solves it by finding an action sequence. In our experiments, a 7DOF robotic manipulator executes the planned action primitives solved by fast downward [120].

PDDL enables the flexibility to control/program a robot to accomplish manipulation tasks. The action primitives get translated to atomic actions realizable via low-level joint control. PDDL combined with category-agnostic affordance detection enables robots to accomplish tasks through the use of non-primary affordances.

4.7 Experimental Results

4.7.1 Dataset and Evaluation Metric

To evaluate the affordance detection and ranking, the standard UMD dataset, covering 17 categories and 7 affordances, is utilized for benchmarking. The UMD dataset is introduced in chapter 4.3 for reader's reference. Note that in the UMD dataset, some object parts admit multiple functionalities. For those object parts, multiple affordances are annotated and the rankings are provided. Additionally, to augment the original UMD dataset with bounding box for training purpose, we segment the background table from the foreground objects, and generate the bounding box ground truth automatically.

To evaluate the performance. we adopt F_β^ω and Ranked F_β^ω for affordance prediction and ranked affordance prediction, respectively. The next two paragraphs describe the details of the two metrics.

1) F_β^ω: The network outputs are a set of affordance probability masks over the input image. Accuracy evaluation involves comparing continuous valued responses against binary valued ground

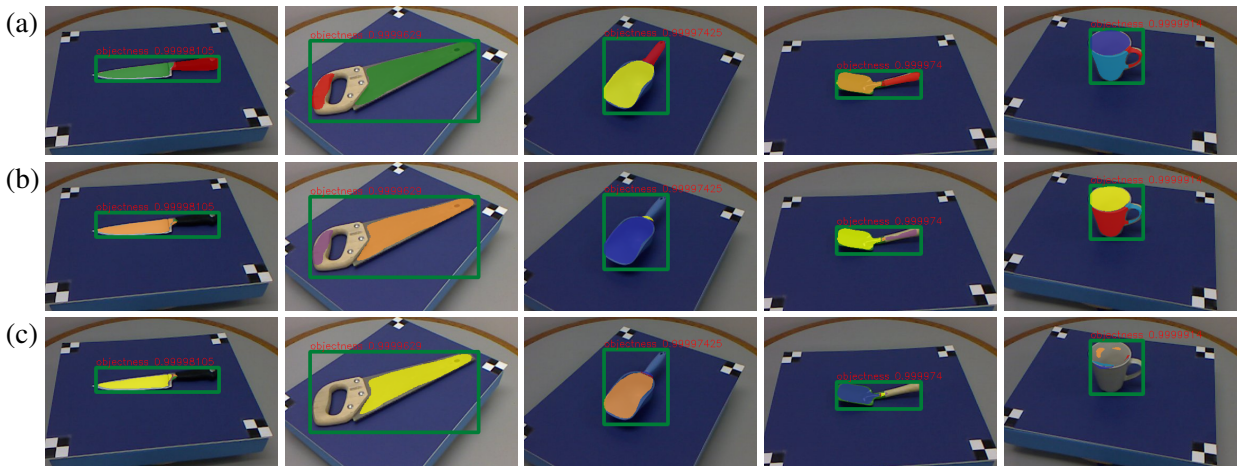

Figure 4.3: Detection results of the method on UMD dataset. The color of mask represents affordance. Red: grasp; yellow: scoop; green:cut; dark blue: contain; blue: wrap-grasp; orange: support; purple: pound. Best viewed in color. **(a)** The first row represents detection results of primary affordances of object parts; **(b)** and **(c)** the second and third rows show detection results of the secondary and third affordances. Note that object parts without coded color indicate all affordance confidences are below threshold and no approate affordance is applied.

truth labels for each affordance. The F_β^ω metric [110] evaluates the probability masks:

$$F_\beta^\omega = (1+\beta^2)\frac{Pr^\omega \cdot Rc^\omega}{\beta^2 \cdot Pr^\omega + Rc^\omega}. \tag{4.6}$$

where Pr^ω and Rc^ω are the weighed precision and recall measures, respectively. Pixels close to foreground ground truth are assigned higher weights.

2) Ranked F_β^ω: The UMD dataset [47] contains ground truth rankings for multiple affordance on object parts. For these, the *rank* weighted F_β^ω [110] is used:

$$R_\beta^\omega = \sum_r \omega_r F_\beta^\omega(r), \text{with} \sum_r \omega_r = 1, \tag{4.7}$$

where ω_r are the ranked weights defined contributing to the weighted sum over the corresponding affordances. The top affordance receives the most weights and so on. Following [110], the weights are:

$$\omega_r = \frac{2^{-r}}{\sum_{r'} 2^{-r'}}. \tag{4.8}$$

Table 4.1: Performance On UMD Dataset (novel category)

	weighted F-measures						
	HMP [47]	SRF [47]	Lakani [51]	DeepLab [56]	Obj-wise Ours	KL Ours	Multi Ours
grasp	N/A	N/A	0.46	0.55	0.61	0.54	0.56
cut	N/A	N/A	0.30	0.30	0.37	0.31	0.35
scoop	N/A	N/A	0.22	0.36	0.60	0.39	0.53
contain	N/A	N/A	0.47	0.58	0.61	0.63	0.60
pound	N/A	N/A	0.08	0.42	0.81	0.55	0.69
support	N/A	N/A	0.03	0.22	0.59	0.75	0.68
w-grasp	N/A	N/A	0.47	0.93	0.94	0.92	0.92
average	N/A	N/A	0.29	0.48	**0.64**	**0.58**	**0.62**

4.7.2 Benchmarking on UMD Novel Objects

Qualitative visualizations of the proposed method are depicted in Fig. 5.6. The performance of the proposed approach on the UMD dataset and comparisons with state-of-the-art methods are presented in Tables 5.2 and 5.3. Note that within existing works, only [54] reports both F_β^ω and ranked F_β^ω scores. HMP and SRF [47] only present performance of ranked F_β^ω. DeepLab [56] represents a strong baseline method for segmentation. We modify it to perform affordance detection. The results demonstrate that the proposed approach outperforms the previous methods on novel categories in both F_β^ω and ranked F_β^ω metrics. The *Multi* indicates a strong variant of the proposed method. *Multi* keeps the same architecture as in [60] except that it has N branches for top N affordances ($N = 3$ in this work). However, in practice, not all object parts have the same number of non-primary affordances. In addition, the network parameter number increase linearly with N in this design, which prevents it from generalizing to large N.

4.7.3 Real-world Manipulation with Detected Affordance

To confirm affordance prediction in practice, an actual robotic arm is deployed to test the performance of the *grasp* and *contain* affordances as in [60]. Comparisons with a state-of-the-art grasp detector deepGrasp [97] and an affordance detector AffordanceNet [60] are shown in Table 4.3.

1) **grasp affordance**: For the *grasp* affordance, unseen objects are presented to the robot for

Table 4.2: Ranking Performance On UMD Dataset (novel category)

	rank weighted F-measures							
	HMP [47]	SRF [47]	VGG [61]	ResNet [61]	Lakani [51]	DeepLab [56]	KL Ours	Multi Ours
grasp	0.16	0.05	0.18	0.16	0.19	0.30	0.32	0.34
cut	0.02	0.01	0.05	0.05	0.18	0.17	0.18	0.20
scoop	0.15	0.04	0.18	0.18	0.28	0.10	0.18	0.21
contain	0.18	0.07	0.20	0.19	0.32	0.22	0.23	0.25
pound	0.02	0.02	0.03	0.02	0.08	0.06	0.09	0.08
support	0.05	0.01	0.07	0.06	0.11	0.04	0.10	0.11
w-grasp	0.10	0.07	0.11	0.11	0.32	0.53	0.52	0.53
average	0.10	0.04	0.12	0.11	0.21	0.20	**0.23**	**0.24**

recognizing object parts for grasping. To execute the grasping action, the grasp frame is determined to be the mean of detected *grasp* pixels, while the orientation of the grasp frame is set by a line fit to the *grasp* pixels.

2) **contain affordance**: Placing a ball in a container evaluates the *contain* affordance. Unseen containers are selected to demonstrate the ability to generalize to novel categories in the wild. The robotic arm is initialized with a small ball held in the gripper. Successful execution occurs if the *contain* region is located and the ball is stably placed into containers.

For the *grasp* tasks with primary affordance (knife and ladle in Table 4.3), the proposed method matches the success rates of the two competitors, while *contain* tasks with primary affordance (mug and bowl) matches AffordanceNet [60] in average success rate. Moreover, the proposed method further expands the opportunities to manipulate objects with *contain* as a non-primary affordance (cup and trowel), which deepGrasp [97] or AffordanceNet [60] may not discover.

4.7.4 Real-world Manipulation with PDDL

To demonstrate further use cases for manipulation tasks, PDDL is adopted to plan a sequence of action primitives for achieving a goal state. Since affordances indicate possible actions to be applied to an object part, detections of objects and corresponding affordances offer necessary information for PDDL to establish an initial state from which to solve for solutions reaching the goal state. The initial state is determined by detected affordances and corresponding locations (computed through

Table 4.3: Physical manipulation

	DeepGrasp [97]	AffNet [58]	Ours	*affordance*
knife	10/10	10/10	10/10	grasp
ladle	9/10	9/10	9/10	grasp
cup	10/10	N/A	7/10	grasp*
mug	N/A	8/10	9/10	contain
bowl	N/A	10/10	9/10	contain
trowel	N/A	N/A	7/10	contain*
average	9.25/10	9.25/10	**8.50/10**	

* indicates the examined affordance is a non-primary affordance, which can only be identified by proposed networks.

Table 4.4: Physical manipulation with PDDL

	affordances involved	primary	non-primary
pliers into container	grasp, contain	9/10	10/10
stablize screwdriver	grasp, support	8/10	8/10
scoop beans	grasp, scoop, contain	8/10	7/10
average		**8.3**	**8.3/10**

point clouds using the Kinect RGB-D sensor). The 7DoF robot arm executes the planned sequence of action primitives.

Table 5.9 summarizes the performance of three manipulation tasks.

1) For the **pliers into container** task, the framework needs to identify the *grasp* affordance of an object for picking up, and recognize an object with the *contain* affordance to drop into. Successful execution results when the the pliers is in a container after all planned actions are executed. In addition to solving for action sequences using primary affordance, we show that the framework can exploit ranked affordances predictions to employ candidate non-primary affordances to complete manipulation (if no primary affordance is available to reach the goal state). For instance, in the *primary* scenario a bowl is presented (bowl possesses *contain* as its primary affordance), while in *non-primary* scenario only a trowel is provided (trowel possesses *contain* as secondary affordance). Out of 20 trials, there is one failure; the pliers slipped from the gripper.

2) The second task, **place screwdriver**, requires the manipulator to *grasp* a screwdriver in the scene, then place it onto an object with the *support* affordance. Successful execution happens when the screwdriver stably remains on the object's *support* surface. In this experiment, a turner

is selected as the target with *support* as the primary affordance; a knife is used in the non-primary scenario since a knife blade has *support* as a non-primary affordance. As the supporting surfaces of the turner/knife do no lie flat, the screwdriver falls off after placement for 2 and 1 of out 10 trials for the turner and knife, respectively. Another failure case with the knife is due to failure to grasp the screwdriver.

3) The last experiment with PDDL is a **scoop beans** task. The framework identifies and localizes *grasp*-able tools with *scoop*-able functionality in a scene, while coffee beans are assumed to be in a *container*. Successful execution occurs when the manipulator grasps an identified tool and scoops beans from the container. A trowel is used for the primary affordance scenario. A bowl is used for the non-primary scenario, showing that it can be identified as a *scoop* tool to achieve the goal state. Among 10 trials with the trowel, 2 failures cases are due to grasping misses of the graspable part. Among 10 trials with the bowl, 2 failures cases are due to missing the container and 1 failure case is due to incorrect localization of the bowl. Note that, for the experiment with bowl scooping, the *scoop* affordance is associated with the *grasp* affordance since a bowl-shape object in UMD dataset is not annotated with the *grasp* affordance.

4.7.5 Detection and Ranking in the Wild

To demonstrate the generalizability, the proposed method is tested on the Cornell dataset. The Cornell dataset involves a total of 885 images with 244 different objects for the purpose of learning valid robotic grasps. Multiple grasp ground truths are labelled on each image. No affordance annotations available for F_β^ω and ranked F_β^ω evaluations. Instead, qualitative results are shown in Fig. 4.5. The model is trained on the UMD dataset with novel-category split, and applied on the Cornell Grasping Dataset.

Compared to image-based approaches [47, 61, 51, 56], ROI-based methods benefit from detection priors and apply to multi-object scenes, as shown in Fig. 4.4. Qualitatively, objectness detection with generalizable affordance segmentation performs reasonably on the multi-object scene. Additionally, though image-based approaches achieve competitive results on UMD benchmark,

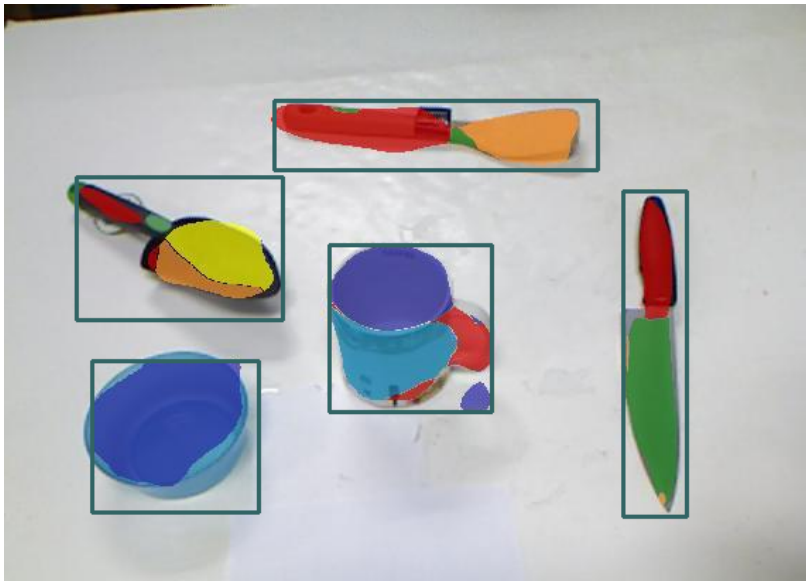

Figure 4.4: Detection results on multiple objects in a manipulator workspace view with the model trained on the UMD dataset. Mask color represents the affordance. Red: grasp; yellow: scoop; green:cut; dark blue: contain; blue: wrap-grasp; orange: support; purple: pound. Best viewed in color.

segmenting multiple objects in a scene with detections of objectness provides necessary information in robotic manipulation tasks. Knowing locations of each object for manipulation in a scene is essential for planning and execution.

Failure cases of the proposed framework on the UMD dataset are presented in Fig. 4.6. Detections of top and secondary affordances are show in Fig. 4.6(a) and Fig. 4.6(b), respectively. A hammer head is detected as *contain*-able with the secondary affordance recognized as *scoop*-able. A similarly incorrect prediction happens on a shears' blade (with label rankings reversed). On the other hand, handles of the spoon and tenderizer are detected as *cut*-table while the secondary affordance is recognized as *support*-able. Since the above two pairs of affordances are highly related in training dataset, incorrect predictions of primary affordances may indicate failure interpretation of the proposed instance mask, and affects predictions of the remaining secondary affordance. Balancing the distributions of co-existing affordances requires further investigation.

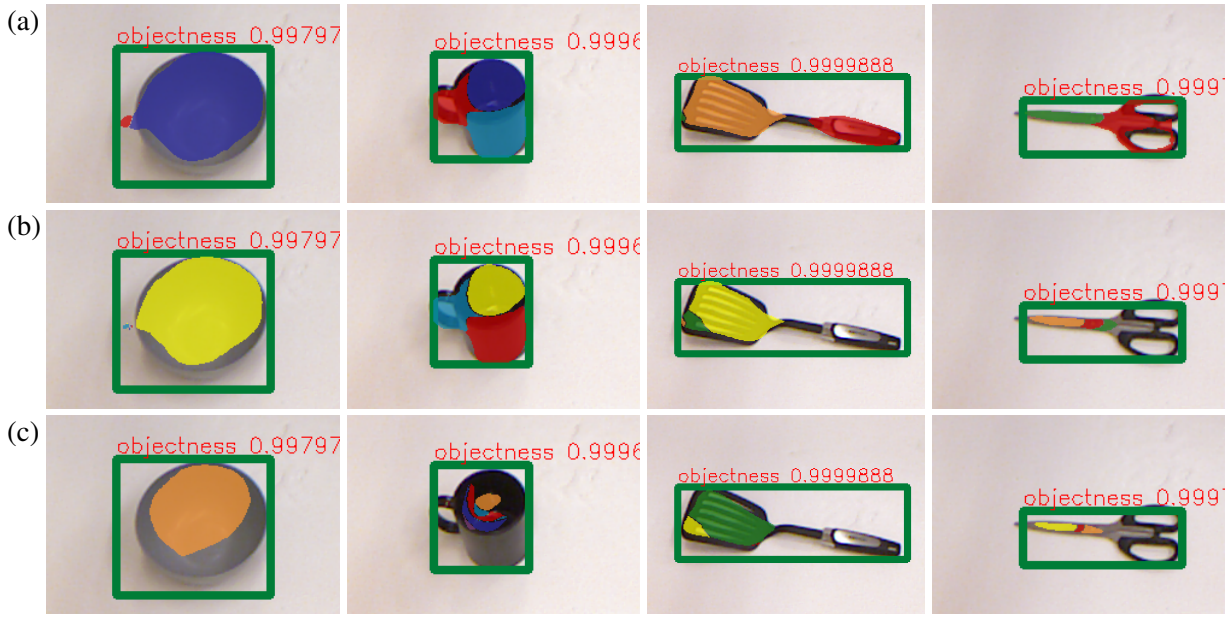

Figure 4.5: Detection results for the Cornell Grasping Dataset. Mask color represents affordance. Red: grasp; yellow: scoop; green:cut; dark blue: contain; blue: wrap-grasp; orange: support; purple: pound. Best viewed in color. **(a)** First row represents primary affordance detection results; **(b)** and **(c)** Second and third rows show detection results of the second and third ranked affordances. Object parts without color coding indicate affordance confidences below the threshold, and no affordance is applied.

4.8 Conclusion

We presented a learning framework to predict ranked affordances of object parts that generalizes to novel categories for real-world robotic manipulation tasks. Compared to previous methods, the proposed framework learns category-agnostic affordances prediction on instance mask proposals, independent of object category priors; with proposed KL-divergence loss, the presented network enables affordance distribution output on a single pixel, allowing predictions of multiple functionalities of an object part. Evaluation includes the UMD dataset with a novel category split for comparisons to state-of-the-art methods, with metrics for both single and multiple affordances. Experiments on physical manipulations showed the proposed framework opens possibilities for manipulating objects with non-primary affordances, while maintaining performance for the primary affordance when compared to state-of-the-art approaches. Physical experiments with PDDL demonstrated the effectiveness of detected affordances for generating action primitives to

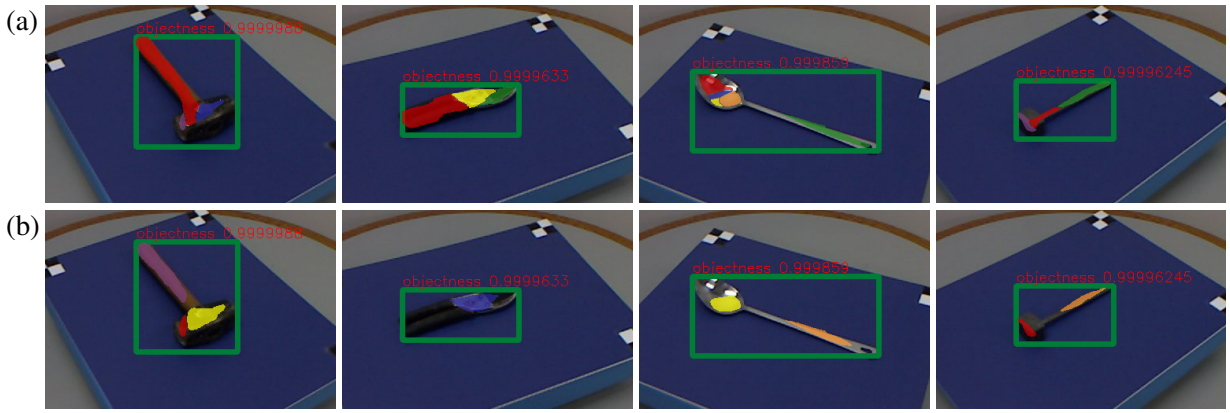

Figure 4.6: Failure cases on UMD Dataset. The first and second rows represent the primary and secondary affordances detection results, respectively. The color of mask represents affordance, color coded as in Fig. 5.6.

accomplish goal-orientated tasks. Although a category-agnostic model for affordance detection is obtained by decoupling the object priors, the category information is sometimes useful in robotic manipulations. One extension in practical use case is to associate the affordance detector with a customized object detector to gain both information. Another improvement could be made by using more representative distribution for KL-divergence loss. Though we achieve good performance in affordance ranking, we also observe the performance loss in top-1 affordance detection. Potentially a better distribution design should lead to performance boost.

CHAPTER 5

IMPROVING AFFORDANCE DETECTION ON NOVEL OBJECTS

5.1 Introduction

In this chapter, we continue the work of detecting robotic affordance for manipulation, with the emphasis on improving performance of per-part affordance segmentation. To be specific, in previously introduced category-agnostic detection framework, an attention module is integrated to the instance segmentation branch for contextual dependency learning. Moreover, an auxiliary task is integrated to the binary classification output to guide affordance learning. In previous chapters, to fully utilize the annotated datasets for unseen categories, an category-agnostic detection framework is introduced to generalize the learned affordances across novel classes. The category-agnostic framework followed the two-stage design for instance-based segmentation outputs, with object priors decoupled. The instance feature benefits object part affordance prediction. However, removing object prior, while enabling generalization of learned affordances, sacrifices segmentation performance.

We observe that, Urban street segmentation [121, 122] has some similarity to robotic affordance segmentation, but there are some critical differences. As opposed to segmentation across the entire image, robotic affordance segmentation of an object involves segmenting a small region with a small subset of affordances. Narrowing the potential subset of affordances based on the object instance context aids segmentation. Therefore, a self-attention mechanism and an auxiliary task of explicitly inferencing existing affordances within a proposalis are considered. We aim to incorporate attention module to the conventional two stage architecture as well as the auxiliary task to capture rich contextual dependencies through the region and explicitly guide affordance learning.

The primary idea of this work is illustrated in Fig. 5.1. We build upon previously proposed

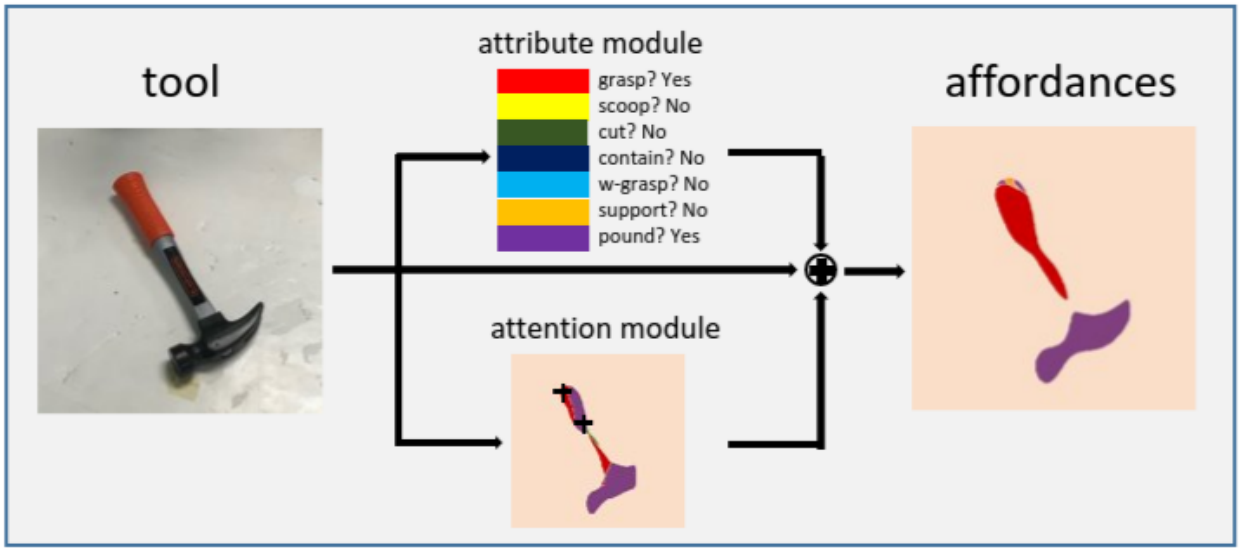

Figure 5.1: Illustration of the affordance detection framework with the proposed attribute and attention modules. The proposed attribute and attention modules improve pixel-wise prediction of object part affordance. The attribute module (upper branch) predicts existing affordances of a region of interest as shareable attributes across categories. This proposed auxiliary task guides the local region feature learning. The attention module (bottom branch) learns dependencies across pixels. For example, the two *plus marks* on the hammer's handle with high correlation should have the same predicted affordance labels.

category-agnostic detection framework. And we show two components to aid the framework to improve instance segmentation of robotic affordance. The bottom branch illustrate the self-attention module for learning dependencies across pixels. The upper branch shows an auxiliary task to guide the affordance learning with explicitly classification of existing affordance in an image patch. Both components show positive effect on improving segmentation performance. We further incorporate the improved category-agnostic affordance detection framework with a pre-trained object detection model, as illustrated in Fig. 5.4, to demonstrate an use case in real-world goal-oriented manipulation tasks with improved PDDL.

Contributions of this work include:

1) A deep network architecture to perform (object) category-agnostic affordance segmentation through a region-based self-attention mechanism. The network, trained with an auxiliary module for multi-affordance classification, achieves the state-of-the-art performance on the UMD benchmark;

2) The multi-affordance classification works as an auxiliary task for affordance segmentation, and is treated as attribute learning approach guides the region-level feature learning to improve performance. An ablation study shows the contribution of the attention mechanism and the auxiliary module;

3) Real-world manipulations with a 7DoF manipulator illustrates the effective affordance prediction for subsequent physical manipulations in different scenarios. Experiments further include task-oriented grasping for cutting and pounding. Lastly, experiments with PDDL-generated action sequence illustrate the use cases in achieving goal-oriented tasks with a object detector and a state keeper.

5.2 Background

Optimizing detection and segmentation boosts the performance with object and localization priors but limits transference of learned affordance labels to unseen categories. One solution is to detect *objectness* of a region proposal instead of predicting object class, thereby enabling category-agnostic affordance segmentation on (object) instance features. The loss of object label constraints on processing requires alternative mechanisms to induce region-driven learning or spatial aggregation. Recent studies on semantic segmentation enhance contextual aggregation by *atrous* spatial pyramid pooling and dilated convolutions [56, 123], merging information at various scales [124], and fusing semantic features between levels [125]. To model long-range pixel or channel dependencies in a feature map, attention modules [126] are applied in semantic segmentation for learning global dependencies [127, 128, 129]. Region-based contextual aggregation in semantic segmentation or object-based affordance segmentation remains unexplored, which this paper aims to address.

One means to induce contextual aggregation is to rely on object attributes. Attributes, as human describable properties, are known to assist vision tasks, such as face detection [130, 131], object classification [131, 132], activity recognition [133], and fashion prediction [134]. Affordances

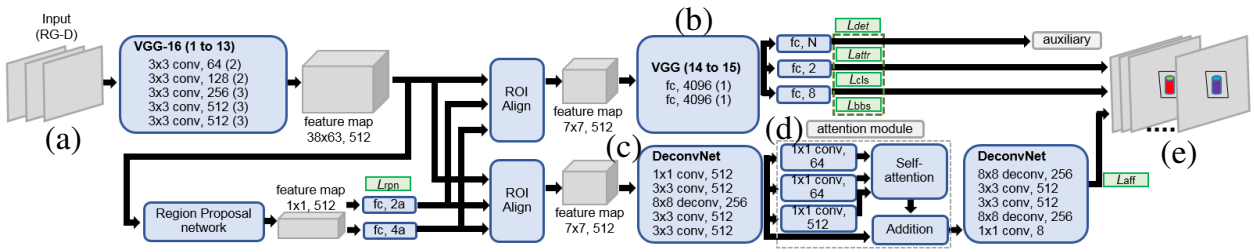

Figure 5.2: Network structure of the proposed detector with self-attention and attribute learning. The network predicts affordances of object parts for each object in the view. Blue blocks indicate network layers and gray blocks indicate images and feature maps. **(a)** RG-D images are input of the network; **(b)** Category-agnostic proposals with *objectness* are forced to predict attributes during training as an auxiliary task; **(c)** Deconvolutional layers lead to a fine-grained feature map for learning long-range dependencies; **(d)** Self-attention mechanism operation is incorporated in affordance branch on the intermediate feature ($30 \times 30 \times 512$); **(e)** The final output includes bounding boxes and multiple layers indicating confidences for affordances on a single pixel.

should also serves as shareable features with semantic meaning whose use could benefit feature learning for object instances. Attribute categories replace the discarded object categories during training to guide feature learning, with the aim of improving generalizability across novel object categories with recognized affordances. We extend [135] by employing attribute prediction to guide instance feature learning while removing object category supervision. The attributes include affordance as a semantic label and additional self-annotated visual attributes. Following [127, 129], we propose to incorporate a self-attention mechanism to model long-range intraregional dependencies.

Different from previous works, the proposed architecture adopts the attention mechanism in the affordance branch and operates on object-based feature maps. Decision dependencies are built upon local regions instead of whole images. The objective of the proposed framework is to guide the instance features with attribute learning, and model the dependencies within the instance feature map for affordance prediction. The affordance knowledge serves to aid real-world robotic manipulation.

5.3 Architecture Overview

This section describes the proposed framework for jointly predicting object regions and their affordance segmentations across novel categories. The general framework of the design, depicted in Fig 5.2, adopts a two-stage architecture [59, 135] with VGG-16 [107] as a backbone. A set of region proposals is collected and input to the detection and segmentation branches for predicting object regions and affordance maps, respectively. To be specific, the shared feature map ($38 \times 63 \times 512$ feature) from the intermediate convolutional layers (layer 13 of VGG-16) are sent to the *Region Proposal network* for region proposals; the two ROI align [59] layers feed the collected instances to the task branches. To generalize the segmentation branch (bottom branch) to novel categories, the detection branch performs binary classification to separate foreground object from background. Segmentation branch takes in category-agnostic object regions for predicting the affordance map within each region. To address the contextual dependencies and the non-local feature learning within a region proposal, we introduce two improvements, described next, to enhance the associations among local features and to guide feature learning to be object aware but not object specific.

5.4 Region-based Self-Attention

The goal of object part affordance segmentation is to group pixels sharing the same functionality and to assign them the correct affordance labels. In urban street semantic segmentation [121, 122], the entire scene usually corresponds to large subset of possible ground truth labels. In contrast, affordance segmentation assigns labels only to object regions, with the assigned labels being a small subset relative to the set of known affordance labels. The semantic context of the image (e.g. a cup) narrows the set of relevant affordances and thus reduces the search space [128].

To aggregate non-local contextual information, the proposed architecture explicitly creates (and consequently, learns) associations between local features of pixels within a region proposal to compensate for the small receptive field of convolutional operations. A self-attention mechanism, as depicted in Fig. 5.3, on the segmentation branch adapts long-range contextual information.

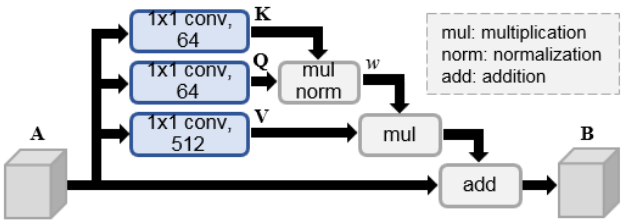

Figure 5.3: Details of the attention module for regional features in affodance branch. **K**, **Q** and **V** refer to key, query and value, respectively

Given an instance feature map $\mathbf{A} \in \mathbb{R}^{c \times u \times v}$, a triplet of *key* $\mathbf{K} \in \mathbb{R}^{c \times u \times v}$, *query* $\mathbf{Q} \in \mathbb{R}^{c \times u \times v}$ and *value* $\mathbf{V} \in \mathbb{R}^{c' \times u \times v}$ feature maps are predicted. We model the contextual relationship w_{ji} with a spatial attention module for features at pixel positions j and i within the instance feature map:

$$w_{ji} = \frac{1}{Z_j} \exp(Q_i^\top K_j) \tag{5.1}$$

where w_{ji} indicates the degree that the i^{th} representation feature impacts on j^{th} feature. Z_j is the normalization term where:

$$Z_j = \sum_{i=1}^{u \times v} \exp(Q_i^\top K_j) \tag{5.2}$$

To aggregate the predicted correlation between features in different position within a region proposal, the feature map $\mathbf{V}$ is associated with contextual relationship w_{ji} and learned a residual function with the original feature map $\mathbf{A}$ for the final output $\mathbf{B} \in \mathbb{R}^{c \times u \times v}$. For features in pixel position j:

$$B_j = \alpha \sum_{i=1}^{u \times v} (w_{ji} V_i) + A_j \tag{5.3}$$

where α is a learnable scale parameter balancing the weighting of global contextual information.

5.5 Affordance as Auxiliary Task and Attribute

Having an object detection branch with binary classification removes contextual information provided by the object category. While it does provide category-agnostic processing, it does not leverage potential information that may be transferable to unseen object instances. Thus, in ad-

dition to predicting *objectness* through object detection, a detection pathway is augmented with a multi-class classification module. The module works as an auxiliary task to explicitly predict existing affordances in a candidate region. The network is trained to predict existing affordances across categories, which transfer to unseen categories during deployment. Predicting pixel-wise affordance map and possible affordances within a region are related. Though the model may learn to predict possible affordances in a region internally, explicitly guiding the learning process as an auxiliary task may allow two tasks affect each other in a positive way [136, 137, 138].

Besides, possible affordances are not solely possessed in training categories, thus predicting existing affordances in a region can be treated as predicting shareable attributes across categories. Attribute learning enhances object classification when suitable attributes are available [139, 140, 141]. In our auxiliary module, affordances are also directly treated as attributes to enhance the learning of visual representation, given that the affordance is recognizable attribute of the unseen categories.

For clarity, below we refer affordance in auxiliary tasks as *attribute*, and retain *affordance* for the pixel-wise segmentation. To guide the feature learning of each region proposal, a task sub-branch parallel to *objectness* detection and bounding box regression is augmented for attribute (affordance) prediction with N outputs, where N is the number of attribute (affordance) defined across categories. Each attribute (affordance) output is a binary classification predicting whether a specific attribute (affordance) is found in the region proposal, based on the instance feature shared with *objectness* detection and regression.

The *objectness* branch identifies foreground from background and hence the class number is $\mathbf{C} = 2$. Let $\rho \in \mathbb{R}^1$ denote the probability of an instance being foreground, $\beta \in \mathbb{R}^4$ denote the corresponding bounding box, and $\alpha \in \mathbb{R}^N$ denote the corresponding probabilities of attributes within the instance region. Define the loss function of complete detection branch ($\mathcal{L}_{\text{det}}$) to be:

$$\mathcal{L}_{\text{det}}(\{(\rho, \beta, \alpha)\}_{c=0}^{\mathbf{C}-1}) = \sum_c \mathcal{L}_{\text{cls}}(\rho) + \lambda_1 \sum_c \delta_{c,1} \mathcal{L}_{\text{bb}}(\beta_c, \beta_c^*) + \frac{\lambda_2}{N} \sum_c \delta_{c,1} \sum_i^N \mathcal{L}_{\text{att}}(\alpha_i). \quad (5.4)$$

where $\mathcal{L}_{\text{cls}}$ denotes the cross entropy loss for *objectness* classification (cls), $\mathcal{L}_{\text{bb}}$ denotes the l_1 loss for bounding box (bb) regression with β_c^* the ground truth annotation, and $\mathcal{L}_{\text{att}}$ denotes the binary cross entropy loss for each attribute. The scalars λ_1 and λ_2 are optimization weight factors, and $\delta_{\cdot,\cdot}$ is the Kronecker delta function.

5.6 Region-based Self-Attention

Both the regional attention module and attribute learning are aggregated in the final network. The attribute learning parallel to the foreground detection works as an auxiliary task to guide feature learning during training; it is discarded during inference. The attention module in the segmentation branch learns a representation of a region proposal gathering rich contextual information; it is applied during inference. To have a higher resolution affordance mask to learn long-range dependencies, deconvolutional layers initially upsample the feature map of the segmentation branch (bottom in Fig. 5.2). Attention is applied after the first deconvolutional operation on the 30×30 feature map, followed by two deconvolutional operations for the final 244×244 affordance map. To compute the loss for the affordance, let $q(x, a)$ denote the predicted affordance mask on a RoI-based feature map, where $x \in RoI$ is the x^{th} pixel in a region proposal, $a \in A$ denote the a^{th} affordance on the pixel. The affordance loss $\mathcal{L}_{\text{aff}}$ is defined as multinomial cross entropy loss:

$$\mathcal{L}_{\text{aff}} = -\sum_{x \in RoI} \frac{1}{|RoI|} \sum_{a \in A} Y(x, a) \log(q(x, a)) \tag{5.5}$$

where $|RoI|$ is the total area of the region of interest, Y is the ground truth of the corresponding affordance mask with A channels.

The overall network inherits Faster-RCNN [96] on a VGG16 [107] backbone with modified detection and segmentation branches while keeping the region proposal network (RPN) intact. Let $\mathcal{L}_{\text{rpn}}$ denote the RPN loss from the original network, the loss for the entire network is:

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{det}} + \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{aff}}. \tag{5.6}$$

5.7 Planning with PDDL

Goal-oriented manipulation may require an agent to achieve a goal state via a sequence of atomic actions, each action involving an intermediate state change of objects/manipulator in the environment. As a standard and widely used planning language, Planning Domain Definition Languange (PDDL) is adopted to encode and generate the desired sequence of actions. The PDDL takes a **domain** definition and a **problem** description. The **domain** pre-defines a list of **predicates** and corresponding effects, while the **problem** includes the initial state and goal state descriptions. To utilize the PDDL, the initial state required by the PDDL algorithm is established via predictions of the proposed framework for objectness and corresponding affordances in the view. Together with the goal state, the planned sequence is solved by fast downward [120] as a list of executable atomic actions for a robotic manipulator.

To handle scenarios where determining an object category is required, incorporating a generally pre-trained object detector enables selecting from multiple tools with similar functionality in the view, improving the flexibility of use cases. To further handle scenarios where an intermediate goal exists, *state keeper* is augmented to the PDDL to save and load the goal state from previous planning outcomes. Such a mechanism is essential due to the defect of both detectors tend to miss an object inside another. As in Fig. 5.4, achieving a large goal while assuring outcomes of a sequence of goals is enabled. Incorporating *object detector* and *state keeper* with the PDDL allows goal-oriented manipulations with tool selections, as well as reusing states for consecutive goals. Goals such as "*grasping an object into a specific container and then grasping the second object into any empty container*" is made possible.

5.8 Vision Evaluation

This section describes the training process and performs benchmarking of the affordance prediction network *AffContext*. Relative to the processing pipeline of a complete implementation on a robotic arm, it describes the perception component and tests performance as a visual processing algorithm

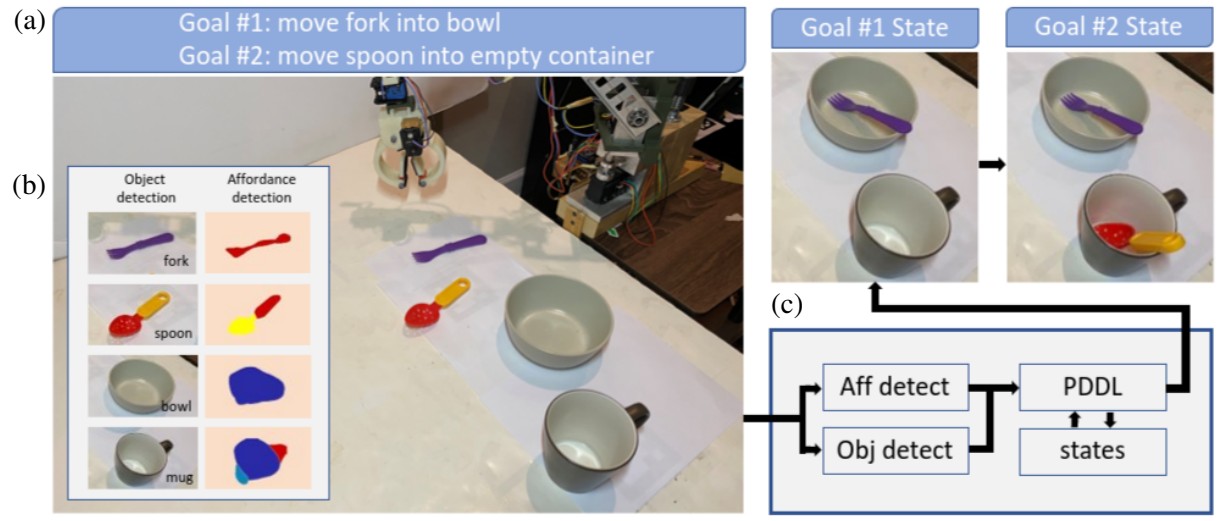

Figure 5.4: Illustration of the agnostic affordance detection framework with PDDL for goal-directed physical robotic manipulations. **(a)** The overall goal is to first move the fork into the bowl, and then move the spoon into the mug. *Goal #1* explicitly specifies the object to grasp (fork), and the object to contain (bowl). *Goal #2*, however, requires knowledge from previously achieved goal state; **(b)** The proposed category-agnostic affordance detector predicts possible actions to be performed on object parts. Together with a pre-trained object detector, both objects and affordances in the robot's view are identified; **(c)** Given a goal state, detected objects and affordances form the initial state for PDDL to plan an action sequence for execution. Given a second goal state, the previous goal state contributes to current initial state for PDDL to inference and plan.

absent the embodied manipulation components. The training dataset is described, as well as the training method, followed by the baseline approaches and benchmarking results.

5.8.1 UMD Dataset

The UMD dataset [47] covers 17 categories with 7 affordances. The categories of objects range from kitchen, workshop, and garden. The dataset contains 28k+ RGB-D images captured by a Kinect sensor with the object on a rotating table for data collection. The segmentation label ground truth is provided in pixel-level, annotating the affordance of each object part. The additional ground truth of object bounding boxes is obtained by filtering out the background table from the foreground objects, and tight the foreground boundary into a rectangle bounding box. The UMD dataset has two benchmarking approaches, *image split* and *category split*. The *category split* is the benchmark that tests unseen categories and is used here for evaluation.

5.8.2 Data Preprocessing and Training

The proposed framework reuses the weights of VGG-16 [107] pre-trained on ImageNet [108] for initialization. The layers for attribute prediction and the affordance branch including the proposed attention module are trained from scratch. To incorporate RGB-D images for geometric information with pre-trained weights, the blue channel is substituted with the depth channel as proposed in [12, 97]. Ideally, any channel can be replaced with the depth channel. The value of depth channel is normalized to the range $[0, 255]$ with 144 as the mean value. Missing value or NaN in depth channel is filled with 0. The whole network is trained end-to-end for 5 epochs. The training starts with initial learning rate $r = 0.001$ which is divided by 10 for every 2 epochs. The training time is around 3 days with a single nVidia GTX 1080 Ti.

5.8.3 Baseline Methods

In addition to including published outcomes for the UMD Benchmarking Dataset, several sensible approaches serve as baselines. They arise from partial or alternative implementation of *AffContext* algorithm, based on earlier work [99]. The first *baseline* approach, labelled *Obj-wise*, is a deep network structure trained with *objectness* detection only, i.e., without regional attention, and without attribute embedding. It is a modified version of AffordanceNet [135] with object category labels unused during training on the *category-split* data. The network only outputs the primary affordance. A second *baseline* approach, denoted *KLdiv* [99], uses the *Obj-wise* network and replaces the cross-entropy loss, $\mathcal{L}_{\text{aff}}$, with KL-divergence for ranked affordance output. Specifically, since each object part is assigned from one to three ranked affordances by human annotators, *KLdiv* treats the ranking of affordances as a distribution and learns to predict ranking during inference time. The ranking output permits secondary affordance of an object part. A third *baseline*, denoted *Multi*, also employs the *Obj-wise* network. To permit prediction of multiple affordances on the same object part, *Multi* simply replicates the segmentation branch to provide a segmentation for affordance (e.g., one-vs-all). The *Multi* segmentation branch grows in direct proportion to the affordance rank quantity (here three). It is a straightforward, brute force method used for com-

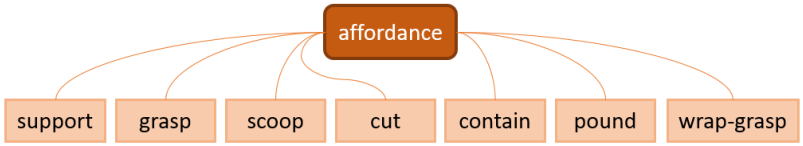

Figure 5.5: Affordances are used as attributes in an auxiliary module for per-candidate region multi-class classification in the UMD dataset.

Table 5.1: Images per Affordance

affordance	images	affordance	images	affordance	images
grasp	18235	contain	7889	wrap	5250
cut	5542	pound	2257		
scoop	2869	support	2317		

parison [99]. Another *baseline* for segmentation employs a modified DeepLab [56, 99]. DeepLab is a widely adopted segmentation network for semantic segmentation, especially for urban street segmentation [74, 122]. It is a representative image-based segmentation approach for comparison. With regards to *AffContext*, the implementation is also modified to permit ranked affordances by using the KL-divergence instead of the cross-entropy loss. However, due to the neural network's memory footprint, only the regional attention mechanism is incorporated (no attribute embedding). We label it $\textit{AffContext}_{KL-att}$.

For the auxiliary task used during *AffContext* training, we treat the original UMD affordance labels as attributes across categories, see Fig. 5.5. In total seven attributes (number of affordance) are defined for representing the UMD tool dataset, as summarized in Table 5.1. An attempt was made to augment these attributes with additional *object attributes* defined in ImageNet and to augment the UMD dataset annotations. *Shape* and *Texture* were selected for their potential relevance (the others are *Color* and *Pattern*). However, no obvious improvements were observed. The resulting weighted F-measures score was 0.67 for these other attribute annotations versus 0.69 for the affordance attribute annotations. All experiments use affordance attributes only.

5.8.4 Evaluation Metric

To evaluate the affordance segmentation responses, derived from probability outputs over affordance classes, against ground truth labels for each affordance, we adopt the weighted F-measures metric, F_β^ω, for the predicted masks:

$$F_\beta^\omega = (1+\beta^2)\frac{Pr^\omega \cdot Rc^\omega}{\beta^2 \cdot Pr^\omega + Rc^\omega}. \tag{5.7}$$

where Pr^ω and Rc^ω are the weighed precision and recall values, respectively [110]. Higher weights are assigned to pixels closer to foreground ground truth. The weighted F-measures outputs lie in the range $[0, 1]$. A second metric evaluates the prediction performance of the rankings for multiple affordance on object parts and applies to the KL-divergence trained network. It is the *ranked* weighted F-measures metric, F_β^ω,

$$R_\beta^\omega = \sum_r \omega_r F_\beta^\omega(r), \quad \text{with} \quad \sum_r \omega_r = 1, \tag{5.8}$$

where ω_r are the ranked weights contributing to the weighted sum over the corresponding affordances [110]. The top affordance receives the most weight and so on, per $\omega_r = 2^{-r} / \sum_{r'} 2^{-r'}$. The ranked weighted F-measures outputs lie in the range $[0, 1]$.

Evaluation results are obtained by running the same evaluation code provided by UMD [47]. All the parameters are the same. There are two parameters associated with ω including σ and α. Specifically, $\beta = 1$, $\sigma = 5$, and $\alpha = \frac{ln0.5}{5}$.

5.8.5 Benchmarking on UMD Novel Objects

The traditional benchmarking scheme for the UMD Dataset is to perform an image-split test, where all object categories are represented. Affordance evaluation is performed only for the known object categories. Top performance for the image-split test lies in the range of 0.733 to 0.799 for the weighted F-measures [54, 135, 56]. The category-split test, where some object categories are

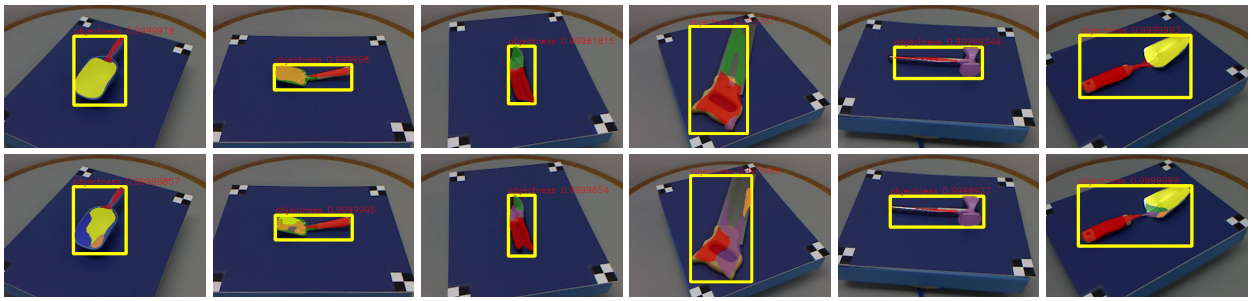

Figure 5.6: Affordance segmentation results on UMD benchmark, where color overlays represent affordance labels, red: grasp; yellow: scoop; green:cut; dark blue: contain; blue: wrap-grasp; orange: support; purple: pound. **Top:** results using *AffContext*; **Bottom:** results using *Obj-wise* baseline.

excluded is a more difficult problem since top performing methods rely on the object category prediction to provide a prior on the potential affordances. The weighted F-measures decrease for the category-split test and there are less reported evaluations. Especially when considering evaluation for weighted F-measures and ranked weighted F-measures. For example, DeepLab [56] performance drops by 34.5% (to 0.48 from 0.733). Since this study aims to explore affordance recognition in the absence of object category knowledge, the category-split test case is performed for single affordance and ranked affordance prediction. Published baselines are included in the benchmarking results when available. When presenting the results, the table contents will be organized according to (i) published results, (ii) strong baselines created in earlier efforts [99], and (iii) the current results for *AffContext*.

Novel Category Affordance Prediction

Qualitative results of *AffContext* are depicted in the top row Fig. 5.6, which has color overlays of the affordance predictions on the objects. The bottom row shows the same for the best performing baseline approach per Table 5.2, which is the *Obj-wise* implementation. *AffContext* improves the consistency of the affordance segmentation as seen by less oversegmentation and more consistent affordance segmentation.

Quantitative evaluation using the weighted F-measure for the UMD benchmark is found in Table 5.2, with available published outcomes for the category-split test. For the multi-affordance *KL-*

Table 5.2: Affordance Segmentation Performance On UMD Dataset (novel category).

	weighted F-measures							
	grasp	cut	scoop	contain	pound	support	w-grasp	average
Lakani [51]	0.46	0.30	0.22	0.47	0.08	0.03	0.47	0.29
DeepLab [56, 99]	0.55	0.30	0.36	0.58	0.42	0.22	0.93	0.48
Obj-wise [99]	0.61	0.37	0.60	0.61	0.81	0.59	0.94	0.64
KLdiv-1 [99]	0.54	0.31	0.39	0.63	0.55	0.75	0.92	0.58
Multi-1 [99]	0.56	0.35	0.53	0.60	0.69	0.68	0.92	0.62
AffContext	0.60	0.37	0.60	0.61	0.80	0.88	0.94	**0.69**
AffContext$_{KL-att}$-1	0.54	0.37	0.42	0.62	0.63	0.87	0.92	**0.63**

Table 5.3: Affordance Ranking Performance On UMD Dataset (novel category)

	ranked weighted F-measures							
	grasp	cut	scoop	contain	pound	support	w-grasp	average
HMP [47]	0.16	0.02	0.15	0.18	0.02	0.05	0.10	0.10
SRF [47]	0.05	0.01	0.04	0.07	0.02	0.01	0.07	0.04
VGG [61]	0.18	0.05	0.18	0.20	0.03	0.07	0.11	0.12
ResNet [61]	0.16	0.05	0.18	0.19	0.02	0.06	0.11	0.11
Lakani [51]	0.19	0.18	0.28	0.32	0.08	0.11	0.32	0.21
DeepLab [56, 99]	0.30	0.17	0.10	0.22	0.06	0.04	0.53	0.20
KLdiv [99]	0.32	0.18	0.18	0.23	0.09	0.10	0.52	0.23
Multi [99]	0.34	0.20	0.21	0.25	0.08	0.11	0.53	**0.24**
AffContext$_{KL-att}$	0.33	0.18	0.18	0.25	0.10	0.10	0.53	**0.24**

div baseline, the top ranked affordance is taken as the affordance output. Hence the label *KLdiv-1*. Likewise the single affordance version of *AffContext*$_{KL-att}$ outputs only the top ranked affordance label (denoted *AffContext*$_{KL-att}$*-1*). *AffContext* has the strongest performance, with the *Obj-wise* baseline next. Compared to the best published result, the proposed approach achieves a 43% improvement (0.48 to 0.69). Compared to the strong baseline, it achieves 7% improvement over *Obj-wise*. Meanwhile *AffContext*$_{KL-att}$*-1* has a 1.5% drop in performance relative to *Obj-wise*, while the non attention version *KLdiv-1* has a 9.4% drop in performance, which stems from an 8.6% improvement of *AffContext*$_{KL-att}$*-1* over *KLdiv-1*. The attribute and attention modules lead to improved performance in the primary affordance segmentation outcomes.

Table 5.4: Ablation Study

	module		**weighted F-measures**							
	attent	attri	grasp	cut	scoop	contain	pound	support	w-grasp	average
Obj-wise			0.61	0.37	0.60	0.61	0.81	0.59	0.94	0.64
Obj-wise		✓	0.62	0.33	0.51	0.61	0.81	0.79	0.94	0.66
Obj-wise	✓		0.60	0.40	0.67	0.60	0.78	0.75	0.94	0.68
Obj-wise	✓	✓	0.60	0.37	0.60	0.61	0.80	0.88	0.94	**0.69**

Novel Category Affordance Ranking

Moreover, the improvements on affordance ranking with ranked weighted F-measures is reported in Table 5.3 . This test is harder, due to the metric heavily penalizing incorrect rankings. It compresses the score output values. The results show that *AffContext* outperforms the existing published approaches and most strong baselines for novel object categories. Compared to the most recent published results [51] and to the strong baseline DeepLab, *AffContext* improves by 14% and 20%, respectively. Evaluation of *AffContext* relative to *KLdiv* and *Multi* shows that *AffContext*$_{KL-att}$ outperforms *KLdiv* and almost matches *Multi*. Recall that the implementation of *Multi* [99] in Table 5.3 is especially designed for affordance ranking by adopting multiple afforance branches for each rank output. Consequently it scales in parameter size linearly with the rank quantity (three). The *KLdiv* approach does not impact the parameter size of the affordance ranking branch (compared to cross-entropy loss). The *AffContext*$_{KL-att}$ approach increases by less than 1% the branch network size. The gap between *KLdiv* (0.2315) and *Multi* (0.2444) is 5.3%. In contrast, for a less than 1% increase in branch parameters, *AffContext* (0.2414) reduces the performance drop to 1.2%. In essence, for a small increase in size the attention module almost matches the performance of the brute force (one-vs-all) network. Note that *AffContext*$_{KL-att}$*-1* outperforms by 1.6% *Multi*, per Table 5.2, for the primary affordance. Thus, the main discrepancies lie with the secondary and tertiary affordances.

5.8.6 Ablation Study

Table 5.4 shows the ablation study results for *AffContext*, where the baseline network is the *Obj-wise* network (first row). The second and third rows quantify the improvements gained from regional attention and attribute learning, respectively. The last row reports the performance with both attention and attribute learning together, which achieves the best result. Each design is independently trained with ImageNet pre-trained weights, instead of finetuing one-by-one. While a 3.7% (0.770 to 0.799) improvement in [135] was regarded as reasonable achievement on UMD image-split benchmark, the ablation study shows 3.1% and 6.2% improvements by introducing attribute learning and ROI-based attention individually. Furthermore, the current gap between the best performing object-aware approach (0.799 for [135]) and the best performing object-agnostic approach (0.48 for [56]) has been reduced from a 40.0% drop to a 13.6% drop and lies close to the lower-end of the range for state-of-the-art object-aware methods (0.733-0.799), e.g., within 6%. The next section, which performs manipulation tests, will further explore how this difference manifests when considering task-relevant manipulation activities, from simply grasping and picking up an object, to performing affordance aware manipulation tasks.

5.8.7 Affordance Detection across Datasets

The last vision-only test demonstrates generalizability across datasets. *AffContext* trained on the novel-category split of the UMD dataset is applied to the Cornell dataset [105]. The Cornell dataset consists of 885 images of 244 different objects for learning robotic grasping. Each image is labelled with multiple ground truth grasps. Though affordance masks are not available for quantitative evaluation with F_β^ω and ranked F_β^ω metrics, visualizations of qualitative results and comparisons are presented in Fig. 5.7. Similar to Fig. 5.6, the outcomes here have more consistent and continuous affordance segmentations for *AffContext* relative to the *Obj-wise* method.

Figure 5.7: Comparison on Cornell dataset. **Top:** detection results of *AffContext* method. **Bottom:** *Obj-wise* model.

5.9 Manipulation Experiments and Methodology

Beyond vision-only benchmarking, embodied robotic manipulation with affordance detection is tested and evaluated. The physical manipulation tests use a custom-built 7-DoF robotic manipulator and a Microsoft Kinect using an eye-to-hand configuration, as shown in Fig. 5.8. In the experiment setting, the sensor input includes depth information. Image areas with target affordances are mapped to 3D space for manipulation planning and execution. Multiple scenarios and possible applications are validated. This section describes the experimental methodology of the five test scenarios, with the subsequent section (Sec. 5.10) covering the results.

5.9.1 Affordance on Seen Categories

As in [135], commonly seen *grasp* and *contain* affordances are first examined. In addition, the *support* affordance is included. All the objects in this experiment are selected from categories in UMD, and thus can be recognized by AffNet [135] (trained on UMD datset). Two instances are picked in each category, where one is similar to instances in UMD, and the other is dissimilar. Examples of selected similar and dissimilar objects are shown in Fig. 5.9. The experiment is mainly designed for benchmarking the proposed category-agnostic detector against the state-of-the-art affordance detection [135], which is trained with object prior but limited to the training categories during deployment.

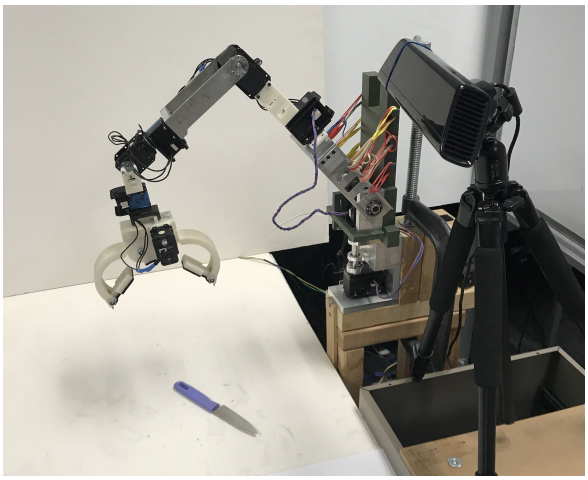

Figure 5.8: Experimental setup for eye-to-hand physical manipulation, with a 7 DoF manipulator and a Microsoft Kinect RGB-D sensor.

For experiment involving *grasp* affordance, the grasp center is achieved by averaging *graspable* pixels, with grasping orientation determined by fitting a line to predicted pixels. Since the camera is set near top-down view, the orientation is then corrected by the relative orientation between the input sensor and working space. For the *contain* and *support* affordance, a small cuboid is placed into (or onto) an object predicted as *contain*-able (*suuport*-able) with the robotic arm. As with the *grasp* affordance, the location is determined by averaging the pixels predicted as *contain*-able (*support*-able). The evaluation metric is discussed in Sec. 5.9.6.

5.9.2 Affordance with Multiple Objects

Real-world scenario usually involves multiple objects in a scene. The workspace may contain task-irrelevant objects. For instance, the robotic arm is required to grasp a knife while a plate and cup are in the scene. The manipulator is asked to put object into the cup while knife and spoon are around. Possible interference may occurs during prediction. Though the proposed method is trained on UMD dataset with single object annotated in each image, the trained model is readily available to detect multiple objects with corresponding affordance maps. As in previous experiment, the commonly seen affordances *grasp*, *contain* and *support* are tested. For objects, the dissimilar set of objects are re-used in this experiment. For each trial, multiple objects (at least 3) including the

Figure 5.9: Examples of similar (top row) and dissimilar objects (bottom row) relative to the UMD dataset. (a) and (d): Most mugs are small and have no or few visual patterns; the dissimilar mug is large and has patterns. (b) and (e): Most spoons large serving spoons; the dissimilar spoon is a small toy. (c) and (f): Most turners are made of wood or steel; the dissimilar turner is made of plastic.

target object (with target affordance) are presented and randomly placed in a visible and reachable area at different locations and orientations.

5.9.3 Affordance on Unseen Categories

While AffNet [135] achieves slightly better vision performance, it is limited to the categories existing in training set (17 categories in UMD). While pixel-wise and bounding box annotation is labor-intensive, generalizability to unseen categories is essential to reuse the learned affordances. Therefore, similar as the above two experiments, commonly seen *grasp*, *contain* and *support* affordances are examined in this scenario. However, all the objects in this experiment are selected outside UMD categories. In this experiment, the proposed model is applicable for novel categories while the AffNet [135] is unable to detect affordance due to its dependency on object detection. The effectiveness of the proposed model on novel categories is examined.

5.9.4 Affordance for Task-Oriented Grasping

Affordances such as *grasp*, *contain* and *support* are usually considered independently, while some affordances such as *pound* and *cut* are commonly utilized in combination with *grasp*. To evaluate those affordances, task-oriented grasping experiments are carefully designed. Task-oriented grasping requires to identify a proper graspable part and functional part in order to accomplish the task. In this section, two task-oriented grasping experiments including *peg into slot* task and *cut through string* task are described.

For pound, a *peg into slot* task is designed as shown in Fig. 5.10a. The peg is half-way through the slot in the beginning and placed at a reachable location. The manipulation is required to detect and grasp the tool, and pound the peg fully into the slot (knock three times for each trial). Same as in experiments involving *grasp*, the end-effector pose for grasping is computed through affordance map and corresponding depth image. The distance between grasping point (where to grasp at handle) and functional point (where to pound on the hammer head) is also computed through affordance map and depth image. The aruco marker on the top of the big cube is for detecting location of the peg.

A second experiment with cut is regarding to a *cut through string* task, as shown in Fig. 5.10b. The string is made by tissues and taped vertically on both ends. The manipulator is required to find and grasp the handle of the tool, and cut the string off horizontally by rotating the wrist of the end-effector (swing three times for each trial). Again, the aruco marker is for detecting the location of the string, while the grasping point and cutting point is computed through the affordance map and depth image.

5.9.5 Affordance with modified PDDL pipeline

The category-agnostic affordance detection generalizes learned affordances to unseen categories by detecting objectness instead of identifying specific object classes. However, knowing object classes may be required in some scenarios such as grasping a tool from a toolbox, or pouring water into a cup next to a bowl. This can be completed by incorporating the category-agnostic affordance

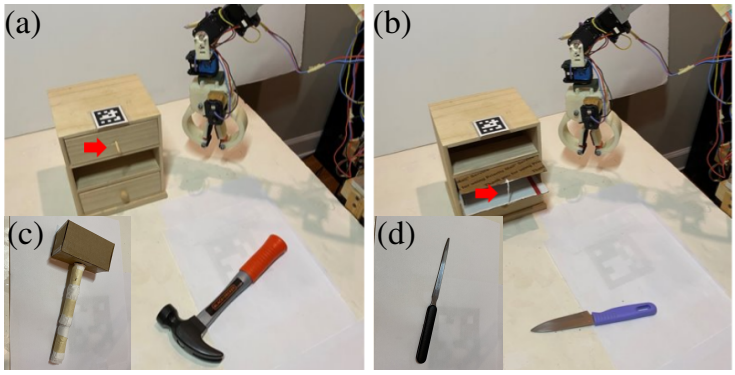

Figure 5.10: Illustration of task-oriented manipulation settings. **(a)** *peg into slot task*: A peg (see red arrow) is half-way through the slot. The manipulator needs to grasp the tool (hammer or tenderizer) and pound the peg fully into the slot. **(b)** *cut through string task*: A string made by tissue is attached vertically (see red arrow). The manipulator needs to grasp the tool (knife or letter opener) and fully cut off the string. **(c)** Custom-made toy tenderizer. **(d)** Letter opener.

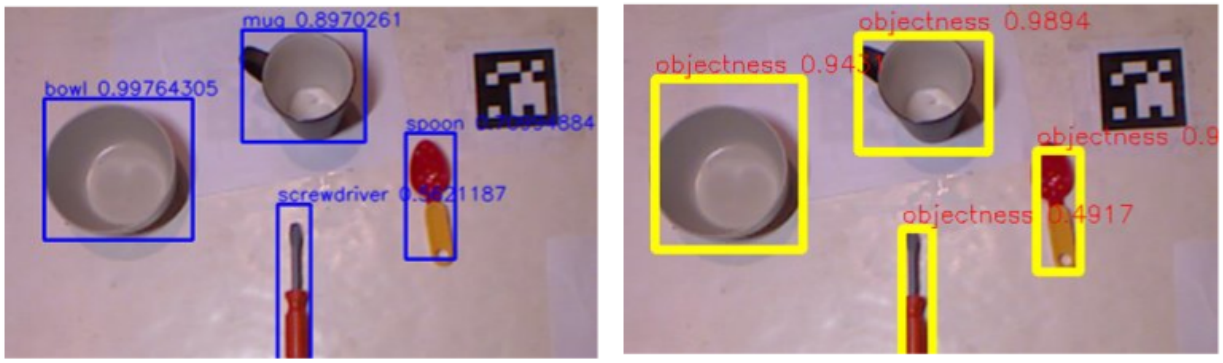

Figure 5.11: **Left:** Candidate bounding boxes (in blue) predicted by a pre-trained object detector. **Right:** Candidate bounding boxes (in yellow) predicted by proposed affordance detector. The detectors are indepdently trained yet predict candidates that are close in locations and similar in size.

detector with a pre-trained object detector. In this experiment, an object detector (faster-rcnn with VGG-16 backbone) is trained on 21 selected classes (tools commonly seen on home/office table). As shown in Fig. 5.11, though the affordance and object detectors are trained independently, the candidate bounding boxes are predicted similarly in our setting. The predictions from two detectors are associated simply by bounding boxes locations and sizes. Apart from identifying objects, to form a complicated manipulation, a straightforward approach is to set a goal state on top of another. And keeping the first goal state to initialize the second goal is essential. Besides, a commonly observed limitation for both detectors is missing objects inside a container, meaning the

state is not able to be established by observing the scene. Therefore, a *state keeper* is incorporated to save and load the previous object states, such as *plate contains spoon* and the location of spoon in the 3D space. In this section, four experiments are carefully designed to evaluate the effectiveness of the incorporated *pre-trained object detector* and *state keeper* as the full pipeline.

The fist two experiments, *pick knife or spoon into bowl* and *select trowel or spoon to scoop beans* examine the object detector to select from objects in the view to accomplish the tasks. To be specific for the first experiment, knife or spoon are randomly selected to be the object in final state, the object detector is required to identify the *grasp*-able object to be either knife or spoon and move into the bowl. Same policy applies to the second experiment.

The last two experiments, *grasp spoon to plate then move to bowl* and *place objects into empty containers* further examines the state keeper to keep track of object state in the view. In order to emphasize the *state keeper*, a container is limited to contain one object in this scenario. The third experiment asks the manipulator to grasp the spoon and move into a container plate in the first phase. In the second phase, the manipulator needs to find the same spoon and move into container bowl. With the *state keeper*, the manipulator loads the state of the spoon to locate its position. The forth experiment involves two *grasp*-able objects and two *contain*-able objects in a scene. In the first phase, one object is chosen to be placed into a container (randomly chosen or specified by name). In the second phase, the second object is required to be placed into an empty container.

5.9.6 Methodology and Evaluation

In this subsection, we describe the different metrics of a success trial in above five scenarios. For each experiment in all five scenarios, we conduct 10 times of trial, and record the outcome as *success* or *failure*. Therefore, the final evaluation result for each experiment is shown as the number of success for each experiment in each of the five scenarios. Details are provided below.

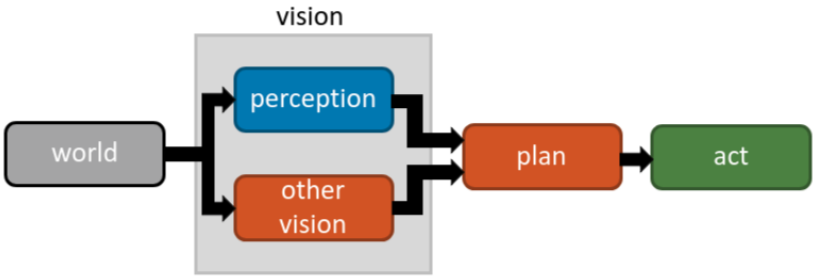

Figure 5.12: Illustration of the physical manipulation process. The vision includes the affordance detection (perception) and other vision processing functions. The visual results are then sent to the planning module followed by physical execution of our robotic manipulator (act module)

Success in Each Scenario

The first three scenarios (Sec. 5.9.1, Sec. 5.9.2 and Sec. 5.9.3) involve simple movements with three common affordances, including *grasp*, *contain* and *support*. A success trial for *grasp* requires the manipulator to stably grasp the target without dropping it. A success trial for *contain* requires the manipulator to put a small cuboid into a target container. Similar as *contain*, A success trial for *support* requires the manipulator to put the small cuboid onto a target supportable surface with falling off. For each experiment, the target is randomly placed in a visible and reachable area at different locations and orientations.

For the fourth scenario (Sec. 5.9.4) involving task-oriented tasks, a success *peg into slot* trial requires the manipulator to first grasp the target without dropping it, and then fully pound the peg into the slot. A success *cut through string* trial is similar, which requires the manipulator to first grasp the target without dropping it, and then fully cut the string with the target. For the last scenario (Sec. 5.9.5) involving modified PDDL, a success trial requires the execution result to be the same as predefined final state (one of the four commands in Sec. 5.9.5). For instance, if the predefined final state is *pick knife into bowl*, a success trial requires the knife to be inside the bowl when execution done; and this requires the manipulator to grasp the knife and place into the bowl.

Failure Categories

To properly evaluate and understand failure sources for the manipulation experiments, failures are divided into one of the three consecutive modules as shown in Fig. 5.12. With reference to the

block diagram, the vision sensor provides visual input to the vision module. Within the vision module, two forms of processing occur. The first is affordance segmentation, here denoted by the *Perception* block. The second is additional visual processing to extract key geometric information for manipulation planning (3D locations, regions, or $SE(3)$ poses). The extracted signals are passed on to the planner (*Plan* block), which uses them to determine the world state and plan an action sequence to reach the target terminal world state. The action sequence get executed in open-loop, e.g, no further visual processing to replan in the case of errors. The manipulator has one shot to complete the task (*Act* block).

Failures can occur in any of these modules. They are broken down into three categories: *perception*, *planning*, and *action*. Affordance related errors are counted towards the *perception* category. To separate the affordance detection error from general vision failures, the visual processing errors that result in bad plans will be counted towards the *planning* category. Additionally, if the planning system generates a bad plan from good information or fails to return a valid plan, the failure counts towards *planning*. Lastly, failures caused by bad physical movements leading to incomplete tasks are recorded as *action* failures.

Common Failure Modes

To simplify the analysis of results, this section describes the failure modes observed across all of the experiments. Several of the experiment scenarios exhibited the same failure modes, thus it is best to describe them in advance. For the affordance or *perception* failure mode, the error sources come in two types: direct and indirect. A direct error is when the affordance prediction is incorrect and fails to identify the target affordance in the scene. Typically, the incorrect affordance is assigned to the region and the target affordance does not appear anywhere. A less common error is the failure to detect the object as an object via the *objectness* classifier, leading to no affordance predictions. In both cases the affordance is not recognized in the scene, and no planning nor manipulation actions can be taken. An indirect error is due to poor or noisy affordance segmentation. The bad segmentation negatively impacts the subsequent geometric reasoning in the other vision processes

and leads to an incorrect plan. As a consequence, the manipulator will fail to complete the task (be it *grasp*, *contain*, *support*, *cut*, *pound*, or *scoop*).

Moving to the vision side of the *planning* errors, failures include incorrect height or length estimation due to depth image noise or bad relative geometry due to incorrect AR tag pose estimates. The other source of *planning* error occurs when the motion planner fails to generate a feasible plan (it happened once). The *action* failure modes include dropping the object while manipulator is moving, shifting of objects within the gripper while being manipulated, or colliding with objects while moving. Though collision could be a function of poor planning, it was common to have this occur once or twice for an experiment indicating that execution uncertainy or variance is the main factor. In-grasp shifts or drops cannot be corrected since there is no continual perception processing to ensure nothing of significance changes during open-loop execution. Larger grasping forces and closed-loop execution would improve on this failure type.

In several cases, the *action* failure is a function of the object geometry. Both the turners and the shovels do not lie flat on the surface nor have trivial geometry. Under normal circumstances a second arm would manipulate the object to present a better relative geometry for placing the object within the affordance action region. Or more tailored placement algorithms could be programmed for these surfaces. In the absence of a second arm and a custom place routine, the robot simply attempts to place the object onto the surface. For the turners and the shovels, the object being placed (a cuboid) sometimes slid off or tumbled off of the surface. Since this investigation does not consider the physics nor dynamics of the manipulation actions, no modifications were made to correct for this occasional failure mode. When describing these error in the experiments, the abbreviated description of sliding or tumbling from the target object will be provided.

5.10 Manipulation Results and Discussion

5.10.1 Affordance on Seen Categories

The proposed method is deployed and compared with state-of-the-art grasp [97] and affordance [135] detectors, with the results provided in Table 5.5. The *grasp average* row indicates perfect

Table 5.5: Manipulation on Seen Categories

	DeepGrasp [97]			AffNet [135]			AffContext			*similar*	*affordance*
Object	Perc.	Plan	Act	Perc.	Plan	Act	Perc.	Plan	Act		
knife	10	10	10	10	10	10	10	10	10	yes	grasp
spoon	10	10	10	10	10	10	10	10	10	yes	grasp
mug	–	–	–	10	10	10	10	10	9	yes	contain
cup	–	–	–	10	10	10	9	9	8	yes	contain
turner	–	–	–	9	9	7	9	9	8	yes	support
shovel	–	–	–	8	8	6	8	8	7	yes	support
knife	10	10	10	10	10	10	10	10	10	no	grasp
spoon	10	10	10	10	10	10	10	10	10	no	grasp
mug	–	–	–	10	10	10	10	10	10	no	contain
cup	–	–	–	10	10	10	10	10	10	no	contain
turner	–	–	–	10	10	9	9	9	8	no	support
shovel	–	–	–	10	10	9	10	10	9	no	support
grasp average	**10**	10	10	**10**	10	10	**10**	10	10		
similar average	**3.3**	3.3	3.3	**9.5**	9.5	8.8	**9.3**	9.3	8.7		
average	**3.3**	3.3	3.3	**9.8**	9.8	9.3	**9.6**	9.6	9.1		

processing and execution for all three methods. The affordance recognition methods successfully identify the graspable regions and use them to lift the objects, thereby matching the performance of the specialized grasp recognition implementation. Moving to the *contain* affordance, the *AffNet* object-aware implementation had perfect visual processing and execution. In contrast *AffContext* experienced a single affordance recognition failure, and two execution failures. A similar trend in the performance loss occurs for the *support* affordance, whereby *AffNet* has 3 affordance failures and 6 subsequent execution failures while *AffContext* has 4 affordance failures and 4 subsequent execution failures. Affordance recognition for *AffContext* demonstrates a less than 3% performance drop over *AffNet* for affordance recognition and execution. These outcomes indicate that the affordance segmentation performance difference between the two methods, when applied to objects known by *AffNet*, does not influence task outcomes as much as the relative F-measures would indicate. For the image-split *AffNet* achieves a 0.8 weighted F-measure, while for the category-split *AffContext* achieves a 0.69 weighted F-measure, which reflects a 13.8% drop relative to *AffNet*.

For the *perception* failures of *AffNet*, 2 were direct (shovel) and 1 was indirect (turner). For *action* failures, all 6 were due to sliding or tumbling (turner or shovel). For *perception* failures of

Table 5.6: Manipulation with Multiple Objects in Scene

	DeepGrasp [97]			AffNet [135]			AffContext			*affordance*
Object	Perc.	Plan	Act	Perc.	Plan	Act	Perc.	Plan	Act	
knife	10	10	10	10	10	10	10	10	10	grasp
spoon	10	10	10	10	10	10	10	10	10	grasp
mug	–	–	–	10	10	10	10	10	10	contain
cup	–	–	–	10	10	10	10	10	10	contain
turner	–	–	–	10	10	8	10	10	7	support
shovel	–	–	–	10	10	10	9	9	9	support
grasp average	10	10	10	10	10	10	10	10	10	
average	3.3	3.3	3.3	10	10	9.7	9.8	9.8	9.3	

AffContext, 2 failures are direct (shovel) and 3 failures are indirect (cup and turner). For *action* failure, 2 failure are due to the end-effector accidentally hitting the containers (cups and mugs) and 4 failures are due to sliding or tumbling (turners and shovels). The main difference lies in the indirect affordance perception failure mode.

5.10.2 Affordance with Multiple Objects

The grasp detector used for grasping comparisons [97] is designed to predict multiple grasps on multiple objects in a scene. On account of this design, grasp selection can be augmented to require choosing a desired grasp from the candidate list (or subset thereof) or to require specifying a region of interest from which the top intersection grasp candidate must be chosen. This experiment adopts the latter modification. The same affordances as in the previous experiment are evaluated: *grasp*, *contain* and *support*. The experimental results are reported in Table 5.6 . For *grasp*, all three methods achieve perfect perception, planning, and execution. Given that *AffNet* is perfect across the other affordances, the discussion looks at them in aggregate. There is a single indirect *perception* failure for *AffContext* for the shovel (*support*). Lastly *AffNet* and *AffContext* have 2 and 3 execution failures, respectively, for a differential of 1. The 2 *action* failures in *AffNet* are due to sliding/tumbling (turner). Likewise, the 3 *action* failures in *AffContext* are due to sliding/tumbling (turner). Overall, *AffContext* shows a less than 2% affordance perception difference and a less than 2% execution difference (for an overall less than 4% task completion difference).

Table 5.7: Manipulation on Unseen Categories.

	DeepGrasp [97]			AffNet [135]			AffContext			
Object	Perc.	Plan	Act	Perc.	Plan	Act	Perc.	Plan	Act	Affordance
screwdriver	10	10	10	–	–	–	10	10	10	grasp
juice bottle	10	10	10	–	–	–	10	10	10	grasp
mouse	10	10	10	–	–	–	10	10	10	grasp
plate	–	–	–	–	–	–	10	10	10	contain
jar	–	–	–	–	–	–	9	8	8	contain
can	–	–	–	–	–	–	10	10	10	contain
griddle turner	–	–	–	–	–	–	10	10	8	support
grill spatula	–	–	–	–	–	–	10	10	9	support
pie server	–	–	–	–	–	–	9	9	9	support
grasp average	10	10	10	0	0	0	10	10	10	
all average	3.3	3.3	3.3	0	0	0	9.8	9.7	9.3	

Table 5.8: Task-Oriented Grasping and Manipulation

	AffNet [135]			AffContext			
Object	Perc.	Plan	Act	Perc.	Plan	Act	Affordance
hammer	10	9	8	10	8	8	grasp, pound
tenderizer	–	–	–	9	8	8	grasp, pound
knife	10	9	9	10	9	9	grasp, cut
letter opener	–	–	–	9	9	8	grasp, cut
seen average	10	9.0	8.5	10	8.5	8.5	
average	5.0	4.5	4.3	9.5	8.5	8.3	

5.10.3 Affordance on Unseen Categories

Given that *AffNet* [135] depends on the object prior and is limited to the training categories, it will not be capable of recognizing affordances for novel objects. Thus, even though it was run for this scenario, the algorithm did not detect the presented objects and could therefore not recognize any affordances. In what follows *AffNet* will not be referenced since all tests failed. The grasp detector [97] is not restricted to specific object categories and can recognize graspable regions of objects. Table 5.7 reports the outcomes for this unseen categories experiment. Again, the *grasp* affordance is perfectly perceived and executed for *AffContext*. The *contain* and *support* affordances have 1 error each (out of 30 trials per affordance class) for a 3% affordance recognition error rate. These errors are indirect affordance perception errors.

When moving from the affordance perception to planning, then to action, there are a total of 4 additional failures to complete the task. The 1 *planning* failure was due to incorrect height estimation (jar). The 3 *action* failures were sliding/tumbling errors (turner and spatula).

Though there is a performance loss relative to *AffNet* for seen objects, the ability to operate without explicit object labels permits more general operation of *AffContext*. In fact, affordance prediction for these unseen objects matched *AffNet* for the seen objects. The consistent task performance for these cases relative to the earlier seen category tests (Tables 5.5 & 5.6), indicates good affordance generalization performance for *AffContext* to unseen objects with known affordance categories.

5.10.4 Affordance for Task-Oriented Grasping

These experiments move beyond *pick-and-place* types of affordance action tasks, which is what *grasp*, *contain* and *support* test. The affordances of *pound* and *cut* require grasping an object and using it to achieve a given goal for some other object in the world. In this case, the other object has a known action region as determined relative to an AR tag (see details in Section 5.9.4). Table 5.8 reports the results from these experiments, which contain tasks with seen and unseen object categories.

For the seen categories *AffNet* and *AffContext* have the same affordance perception performance, however *AffContext* has one more planning failure than *AffNet* (3 vs 2). The *planning* failures in all cases are due to visual processing errors (bad heights). Compared to previous scenarios, the two designed task-oriented grasp tasks in this scenario require good estimation of the height of the peg and string. Apart from the AR tag in the work space for estimating relative transformation between camera and manipulator, another AR tag in attached on the target object with peg or string as shown in Fig. 5.10. Due to the noisy depth and errors introduce from both AR tags, the returned height estimation of the *pound* or *cut* position are sometimes not accurate enough and lead to misses. The *action* failure for *AffNet* was due to the hammer tilting after being grasped.

Moving to the unseen objects, the self-made tenderizer (see Fig. 5.10c) case had an indirect

Table 5.9: Manipulation with Object Detector and State-PDDL.

Activity	Success			*Affordance*	*Detect.*	*State*
	Perc.	Plan	Act			
pick knife or spoon into bowl	10	10	10	grasp, contain	✓	
select trowel or spoon to scoop beans	10	10	8	grasp, scoop, contain	✓	
grasp spoon to plate then move to bowl	8	7	7	grasp, contain	✓	✓
place objects into empty containers	9	8	8	grasp, contain	✓	✓
average	9.3	8.8	8.3			

perception failure for the *grasp* affordance plus a *planning* failure (bad height). For the letter opener (see Fig. 5.10d), there was one direct *perception* failure for the *grasp* affordance and an *action* failure. The letter opener tilted after the first cut attempt so that the two additional attempts could not succeed.

The affordance recognition for *AffContext* in this set of experiments is close to that of the earlier experiments but slightly worse. There is a 5% error rate at the affordance level with a further 12% drop when translating to action. This increased performance drop from perception to action is a function of more perception-based measurements outside of the affordance category. They accounted for 4 of the 5 post-*perception* module failures, or 80%. In contrast the earlier experiments have an aggregate *planning* failure rate ten times lower, at 8%. The current processing schemes are not robust to non-affordance visual processing errors. Nevertheless, these results demonstrate the power of combining affordance reasoning with symbolic reasoning to plan and execute manipulation activities.

5.10.5 Affordance with Modified PDDL Pipeline

Including state memory of prior actions to support contemporary vision failures associated to overlapping or occluding objects supports additional experiments that have goal state specification with multiple, feasible solution plans. The experiments, detailed in Section 5.7, led to the outcomes reported in Table 5.9. In our fist experiment *pick knife or spoon into bowl*, the PDDL has a goal state to be either *bowl contains knife* or *bowl contains spoon*, while both knife and spoon are in the view with a bowl. The object detector confirm the *grasp*-able object to be either knife or spoon

and *contain*-able object to be bowl when generating the initial state from visual information. The planned sequence of action primitives is executed for physical manipulation. A similar experiment *select trowel or spoon to scoop beans* is performed, where both a *scoop*-able trowel and spoon are present in the view to scoop coffee beans. With the incorporated object detector, the flexibility of choosing between tools to achieve a task is improved. For both of these experiments, the object detection and affordance recognition modules worked perfectly. The 2 *action* failures for the second experiment were due to hitting the edge of the bowl while performing the *scoop* action primitive.

The third and fourth experiments require the state memory component to complete the specified task. During successful operation for the third experiment, the system correctly identifies the empty container with the *state keeper* as well as when the object of interest is placed on the container (e.g, the plate). The 2 *perception* failures for the third experiment, *grasp spoon to plate then move to bowl*, were direct and caused by the *objectness* of the plate not being detected by the affordance detector. The 1 *action* failure was a motion planning failure. For the fourth experiment, *place objects into empty containers*, the system keeps track of the state of each container in order to accept different combinations of the goal state. The 1 *perception* failure is direct; the system did not perceive the *contain* affordance for the mug. The 1 *planning* failure was due to the mug not being detected on the object detector side (it is not an affordance related error). The *perception* error rate for this set of experiments ($\sim$7%) is a bit higher than for the previous ones in aggregate ($\sim$3%), but is within the margin of error given the sample size. Meanwhile, the execution success rate drop of 10% is consistent with the previous Task-Oriented experiment. This experiment demonstrates that the *AffContext* perception module and PDDL planning module provides the flexibility to specify and execute goal-oriented manipulation tasks based on affordance informed state information about the world.

5.10.6 Discussion of Aggregate Performance

Looks like overall performance is around 97% for affordance recognition in simple scenarios. Followed by 92% task completion success. The 3% drop in affordance recognition could be improved

through better network design with regards to the overall detection and recognition processes, while the following 5% performance drop could be improved through better segmentation (more consistent regions) and improved awareness of the scene geometry (e.g., orientation of support surfaces). The model in the physical experiment is trained on the UMD dataset. The domain shift between vision dataset and workspace potentially causes performance drops. Though we showed perfect estimation of affordance map is not a requirement for success in execution, better affordance segmentation result could be achieved by finetuning on a small set of workspace or task-relevant data before deployment.

In addition to simple manipulation tasks with affordance-based pick and place action demands, the task-oriented manipulation or sequential manipulation experiments reflects moderately complex scenarios requiring additional geometric processing to follow through on the task be exploiting a part affordance. Affordance perception for these demonstrated a 94% success rate with *AffContext* when aggregated, while execution led to a final success rate of 83% (for an 11% drop from perception to action).

In aggregate, across all experiments, affordance recognition performance is 96% and task completion is 88%. Our earlier work has collected grasping success rates for various deep learning algorithms (see [97, 142]). For vision only grasp detection, state-of-the-art success rates vary from 87% to 97%. For embodied grasping based on these algorithms, success varies from 80% to 97% for the simple case of grasping. The success rates of the affordance perception module lies at the upper end of the vision-only success rates but considers more complex affordance cases. The success rates of the simple affordance-based manipulation tasks lie at the upper range of the embodied grasping range. Meanwhile, the success rates of the various, more involved manipulation activities tests lie at the lower end of this range. However, the larger gap is a function of errors in other parts of the perceive, plan, act pipeline. These outcomes can be improved by improved programming of the associated support perception and planning modules. Across the experiments, 6 of the 7 affordances in the UMD dataset were tested with the missing one being *wrap-grasp*. Considering the tasks involved, affordances employed, and the outcomes achieved, *AffContext* exhibits state-of-the

art performance for the affordance perception pipeline and the perception to action pipeline.

5.11 Conclusion

This paper described a novel framework to predict affordances of object parts in an image. The deep network framework learns to generalize affordance segmentation across unseen categories in support of robotic manipulation. Compared to previous approaches, this framework performs affordance segmentation within predicted foreground object proposals. The framework learns a self-attention mechanism within the proposed foreground region and selectively adapts contextual dependencies within each instance region. To compensate for the absence of object category priors, category attributes are incorporated to guide the feature learning. Evaluation on the UMD dataset used the novel category split for comparison to state-of-the-arts, including several image-based and region-based baselines. Experiments with physical manipulation demonstrated the effectiveness of this framework for manipulating unseen object categories in the real-world. For future directions, one important component is to incorporate the closed-loop feedback for robustness. We observed that some failure cases could be avoid with visual feedback, such as object dropping during moving the end-effector.

CHAPTER 6

APPLICATION: ASSISTIVE MANIPULATION IN HUMAN-IN-THE-LOOP SYSTEM

6.1 Introduction

For a robot agent to physically interact with the real-world, being able to manipulate objects is an essential step. Though it is relatively easy for human, reliably grasping arbitrary objects remains challenging for robots. The advanced ability to manipulate benefits the applications of robotics in industrial use cases, such as part assembly, binning, and sorting. Likewise, it would advance the area of assistive robotics, where the robot interacts with its surroundings in support of human needs. In this chapter, we show our previously introduced grasp affordance detection integrated into a human-in-the-loop system for assistive manipulations.

Robotic manipulation with assistive arms has been adopted to help elder generations and people with paralysis in upper limb to accomplish daily activities. Based on the analysis of the International Classification of Functioning, Disability and Health (ICF), an able-bodied individual conducts 3,964 activities in five days [143]. Commonly used assistive arms with traditional manipulator control interfaces can be improved with shared/improved autonomy through a human-in-the-loop system with advance vision algorithms and adequate interfaces.

The proposed hands-free control framework of an assistive manipulator involves utilizing grasp affordance detection to shift control burdens from users to machines. In the meanwhile, as shown in Fig. 6.1, two input components are integrated for improvement of autonomy: the Tongue Drive System (TDS) and the augmented reality (AR) glasses. The overall objective is to demonstrate the applicability of the vision algorithm with AR interface in increasing the autonomy so as to bridge the ability gap in assistive technology.

The primary idea of this work is illustrated in Fig. 6.1. The modules on the top row include the AR glasses menu and the TDS. Both modules serve as a human-robot interface for the human

to select summarized intents on the menu via the TDS. The selected intents will be executed by an assistive manipulator shown at the bottom right. The vision system at the bottom-left involves our grasp affordance detection. With the grasp detection, the system automatically guides the assistive robot to the pre-grasp pose, without human-involved Cartesian control or direct motor control. We show the autonomous grasping powered by grasp detector and human-robot interface effectively enables a full autonomy human-in-the-loop control for the pick-n-place task. Compared to Cartesian control, the system is 4 times faster with small decrease in task accomplish rate. Beyond the standard pick-and-place task, we show our system is suitable for more manipulation tasks such as drawer opening, shaking salt and unscrewing cap.

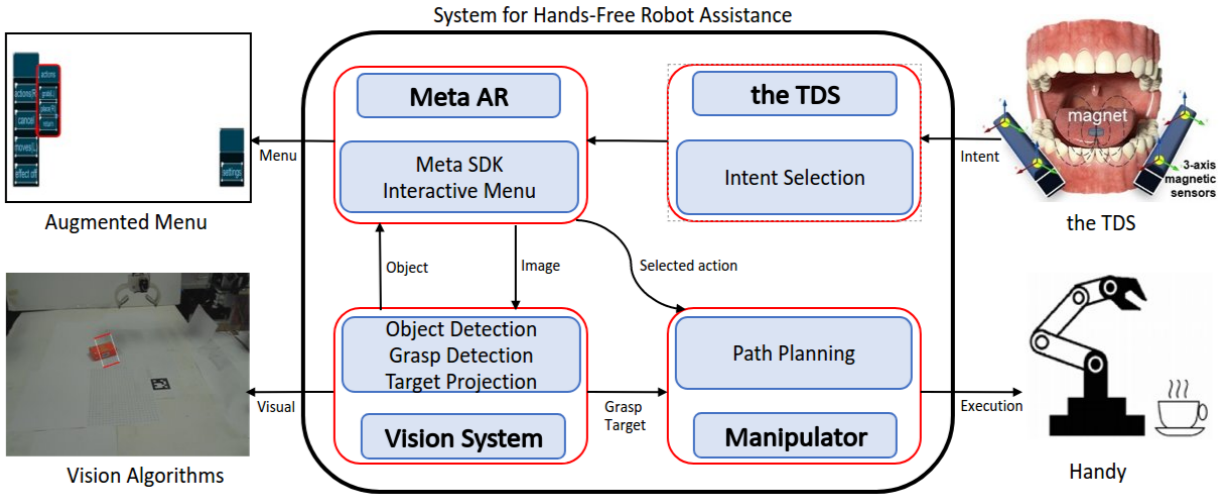

Figure 6.1: Block diagram of data flow for proposed system modules. Top-left: AR glasses recieves RGB-D images; bottom-left: vision system performs object detection, localization and grasp detection; up-right: TDS receives user's input and triggers robotic arm; bottom-right: a 7-DOF robotic arm performs manipulation based on human intent.

Contributions of this work include:

1) A streamlined framework to enable human-robot collaborative manipulation tasks in a hands-free manner. It leverages the users egocentric perspective, summarizes high-level intents, and enables autonomy to minimize user effort in assistive systems;

2) An architecture integrating modern perception algorithms to simplify complex, low-level processes to high-level user commands for minimizing user effort;

3) Quantified results demonstrating reduced user commands, improved task completion time, and a competitive success rate for several manipulation experiments, all made possible through increased autonomy.

6.2 Background

In the United States, paralysis afflicts 5.5 million people [144]. To accomplish daily activities, people with high-level paralysis rely on caregivers and/or environmental modification. Personal mobility devices and environmental control systems provide some degrees of autonomy [145, 146], yet there remains a gap between the activities afforded by these interventions and the needs of the paralyzed population. Research and translational efforts in robotics and assistive technologies (AT) indicate that these emerging support technologies can bridge the existing ability gap.

Assistive manipulations with robotics have long been considered as enabling technologies for self-supportiveness and independence in accomplishing Activity of Daily Livings (ADLs) [147, 148, 149], for populations including the elders and paralyzed people in need. Assistive robotic arms such as the JACO arm and the MANUS have 6-7 degrees of freedom, and admit execution of many ADLs. However, even for non-paralyzed populations, traditional manipulator control interfaces such as Cartesian control or direct motor control, require some levels of expertise and involve human operation error [150]. It is challenging for people with paralysis of arms to fully control an assistive system at the required proficiency level [151, 152]. With the aid of vision algorithms such as detection and localization, the control effort could be gradually minimized with the increase of system autonomy. Increased robot autonomy improves performance [147, 153, 154, 155]. To harness the robot's autonomous capabilities, effective interfaces for communicating human intent to the robotic arm are essential. Users with paralysis should ideally interact with an easily accessible hands-free interface [156].

To harness the robot's autonomous capabilities, effective interfaces for communicating human intent to the robotic arm are essential. Users with high-level paralysis should ideally interact with an easily accessible hands-free interface [156]. For user input, the proposed system employs a

Tongue Drive System (TDS); it is a wireless assistive technology for translating tongue motion to discrete commands [157, 158, 159]. Studies show that it is an effective hands-free interface, with higher throughput and accuracy as compard to other devices such as EEG, EMG [160], eye tracker, and Sip-and-Puff The TDS requires shorter training and calibration times (under 5 minutes); users learn to interface the TDS quickly. Further, the tongue muscle has a low rate of perceived exertion and does not fatigue easily.

Lastly, robot communication to the user should be easy and non-disruptive. Visual display devices with dynamic menuing provide the desired flexibility and compatibility with the TDS interface. Display device options include laptops, tablets, audio assistants, and augmented reality (AR) glasses [160, 161]. An AR application to the rehabilitation and assistive systems field includes [162], where the AR system provided feedback on prosthetic hand grasping quality.

With the TDS as user input and the AR glasses as an interface, this chapter illustrates a human-in-the-loop system integrating grasp affordance detection for improving system autonomy.

6.3 System Architecture

This section describes the human-robot collaborative system, with Figure. 6.1 depicting the structure of the human-in-the-loop system. There are two main sub-systems: the autonomous robot (bottom row of blocks) and the human interface (top row of blocks). Due to the expectation of hands-free operation and the need for user guidance of the robot's actions, the coupling of the two systems is essential to closing the perceive-plan-act loop. On the perception side, augmented reality (AR) sensors provide visual input to the *Vision System* block consisting of RGB-D images. After interpretating scene, it generates a corresponding virtual menu of actions for the manipulator to execute (*META AR* block). Once the AR presents the virtual menu to the user, it waits for the user's intent as feedback, triggered via the Tongue-Drive System (*the TDS* block). The TDS input modality enables hands-free operation by mapping tongue movements to button press operations for virtual menu selection. The selected intent will trigger the manipulator to plan and autonomously complete tasks (*Manipulator* block).

6.3.1 Interface: Egocentric Vision through AR glasses

The AR system, a META-1 Developer's Kit, bridges the information gap between the user and the robot through bi-directional transmission of visual information. META glasses are equipped with a color camera, a time-of-flight depth sensor, and an IMU. The AR projected field of view is 23 degrees with 960×540 resolution for each eye. Using AR to visualize actions and provide context-based menuing systems tends to be more efficient and intuitive when compared to other modalities [163]. Specifically, providing information to the user in their field of view does not require gaze to be broken from the object of interest. Since the AR system has visual sensors that provide the ego-centric view to the controlling computer of the robotic arm, the robot has a similar visual perspective as the user. A processed scene would recover objects of interest matching the user's field of view. As shown in Figure 6.1a, there is a AR menuing system for detected objects. The interface is a Unity3D canvas with interactive buttons. Illustrations of the menu design are shown in Figure 6.2. The main menu is shown in Figure 6.2a, where one could see several bottoms with text on them. Note that, to avoid the users to trigger experiment unrelated bottoms, some bottoms are disabled. To be specific, only the *actions* and *moves* are functional in the main menu during the experiments. The letter *R* behind the *actions* indicates that the users is able to trigger the *actions* bottom by moving their tongues to the *R*ight. Same logic applies to the *moves* bottom, where the bottom is triggered by by moving their tongues to the *L*eft. Overall, letters *L*, *R*, *U*, *D* indicate *Left*, *Right*, *Up*, *Down*. Only the bottoms with these four letters are functional. Figure 6.2b shows the sub-menu after the *moves* bottom is triggered in the main menu. In the sub-menu, one could see four functional bottoms for the users to select (forward, back, left, right) which will be used for task 0 in Section IV.C. Note that, once the sub-menu is popped out, the tongue movement maps to the sub-menu not the main menu (*R* maps to *right* instead of *actions*). Figure 6.2d shows the the sub-menu after the *actions* bottom is triggered in the main menu. Only two functional bottoms in this sub-menu, which will be used for task 1 and task 2 in Section IV.C. There is a visual effect indicating the bottom is triggered, as shown in Figure 6.2c. The selected bottom turns to green and turns back to its original color.

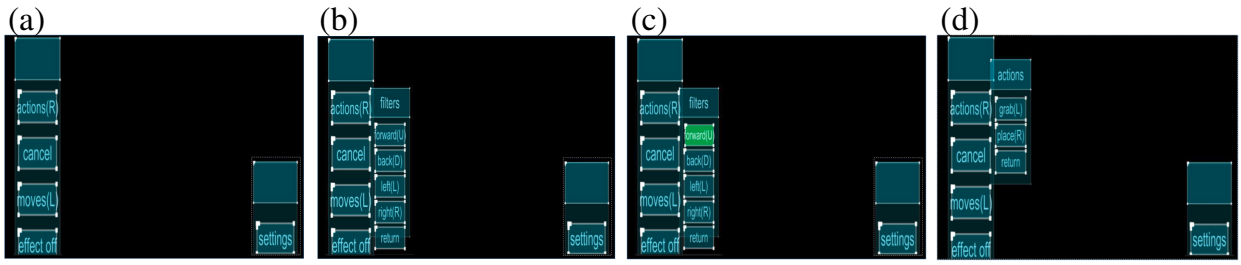

Figure 6.2: Illustration of META menu design: (a) the main menu; (b) the sub-menu after pressing *moves* in the main menu for *task 0*; (c) the visual effect when a bottom is being triggered (in this case, the *forward* command is triggered); (d) the sub-menu after pressing *actions* in the main menu for *task 1* and *task 2*.

6.3.2 Interface: The Tongue-Drive System (TDS)

Button selection in the menu involves the TDS (Figure 6.1c). The TDS headset contains five magnetic sensors (2 near each side of the cheek, 1 on the top of the head) held by the custom 3D printed components affixed to the AR headset, as seen in Figure 6.3. These sensors are used to locate the position of a disk magnet (D21B-N52, K&J Magnetics, Inc.) temporarily attached to the tip of the tongue using tissue adhesives, as shown in Figure 6.4b. Communication with the computer is through Bluetooth Low Energy. Prior to use, the TDS requires a calibration stage to remove the effect of the external magnetic field (EMF). Then, an RBF SVM model is trained to detect the the position of the disk magnet. Once calibrated/trained, the TDS detects tongue movement at 4 locations relative to the rest state of the tongue, denoted as *left*, *right*, *up* and *down* in Figure 6.4c. The movements map to button presses in the AR menu for selections.

6.3.3 Visual Interpretation of the Users Environment

The *Vision System* is essential in linking the human and the robot. It informs the virtual menu interface provided to the user and consists of available manipulator actions. In addition to identifying object types and affordances, it assists manipulation tasks by establishing an object's pose and predicting candidate grasp strategies. These processes pass motion planning information to the *Manipulator* block.

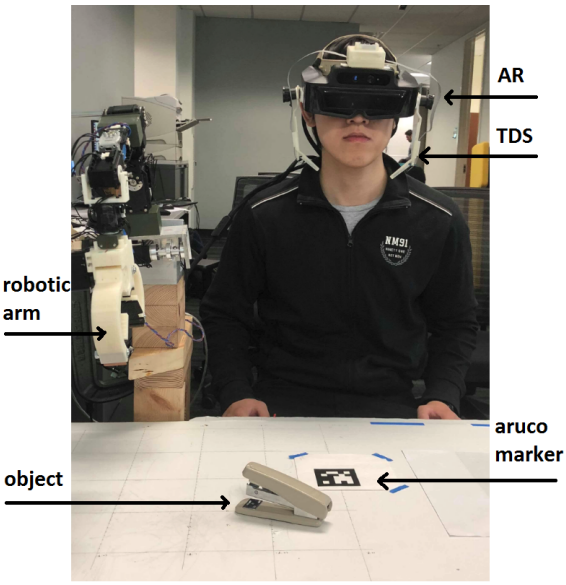

Figure 6.3: Illustration of an user wearing the AR glasses and the TDS in the proposed assistive system with an assistive arm.

object detection

The vision system uses a state-of-the-art deep neural network architecture, YOLO [164], to recognize objects in a scene. YOLO consists of 24 convolutional layers followed by 2 fully connected layers. YOLO's simpler pipeline and unified architecture results in fast visual processing rate; instances of YOLO achieve more than 150 fps, which is suitable for applications with real-time requirements. For better performance regarding the intended application, the network is pre-trained on the PASCAL VOC 2007 train/val + 2012 train/val datasets. Fine-tuning uses a manually collected dataset of objects. The batch size is set to 64, and learning rate is set to 10^{-3} and decreases by 10 at every 10 epochs. Figure 6.5a shows the typical output of the YOLO component.

object localization

Manipulation requires object spatial location relative to the manipulator. To simplify the overall system, the manipulator base is assumed to be fixed. An ARUCO marker [165] defines the world origin and is presumed to be visible along with the objects of interest (see Figure 6.3). Upon seeing the ARUCO marker in the field of view, the system registers the camera frame relative to the ma-

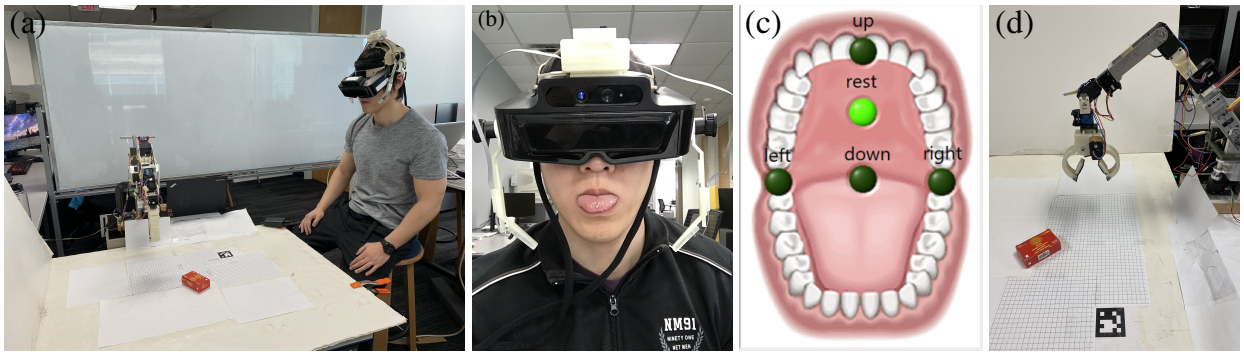

Figure 6.4: (a) Experimental setup of the assistive system for manipulation tasks. (b) User wearing the AR glasses and the TDS. A magnetic disk is attached to the tip of the tongue. (c) Visualization of four positions relative the rest pose of the tongue. (d) The assistive arm with 7 DOF for manipulation tasks.

nipulator base frame. Additional processing of the 2D object bounding box output from the object detection stage leads to the 3D bounding if the object in the manipulator base frame. Region growing segmentation applied to the object's cropped point cloud, followed by extraction of the largest cluster, and removal of the table surface points generates the object point cloud. Its 3D bounding bounding box localizes the object for manipulation purposes. Figure 6.5b represents a processed output of 2D bounding boxes in Figure 6.5a. The block diagram in Figure 6.7 summarizes the pipeline.

graspable locations

A second deep neural network architecture recognizes graspable locations for robotic manipulation from RGB-D inputs; details are in [97]. The grasp configuration output by the network is a 5D grasp rectangle representation, $g = \{x, y, w, h, \theta\}$, applicable to parallel plate grippers. As shown in Figure 6.6, the coordinates (x, y) are the center of the rectangle, θ is the orientation of the rectangle, and (w, h) are the and height; typical candidate grasp of objects are presented on the right. This network outputs a list of grasp candidates with a 5D grasp rectangle representation and corresponding confidence score to inform the manipulator planning.

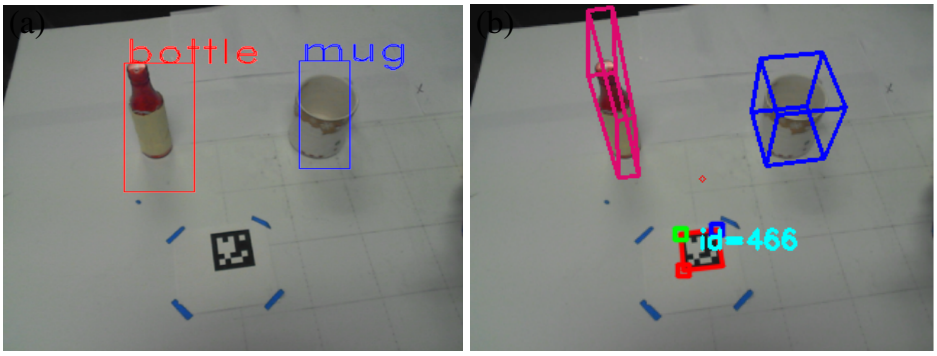

Figure 6.5: Object detection and localization for visual interpretation: (a) a state-of-the- art deep neural network YOLO is finetuned on desired object classes to detect object of interest in 2D image input; (b) regions of interest in 2D and associated point clouds are processed for 3D bounding boxes with respect to an ARUCO marker for manipulation tasks.

6.3.4 Planning for Autonomous Manipulation

The robotic arm is a 7-dof redudnant manipulator shown in Figure 6.4d. Path planning for manipulation is performed via a modified MoveIt! package in ROS. The modification admits path planning with mixed initial and final configurations [166, 167]. The former are given as joint angles and the latter as end-effector configurations, thereby avoiding the need for inverse kinematics. The grasping task relies on the object location and the approaching direction as estimated by the vision system, which are input to the manipulator path planner. Once a human intent is triggered, the manipulator autonomously completes the tasks without detailed interactions such as Cartesian movement of the end-effector or specifation of grasp poses.

6.4 Experiments and Methodology

Evaluation of the hypotheses for the shared autonomy approach to hands-free robotic manipulation assistance involved executing human subjects research with several test tasks for each participant. There is a study involving recruited human subjects with no prior experience using the system (*novice users*) and a study involving an *expert user* of the system. The calibration and setup steps for the two populations was the same. They are described first. Next, the overall flow of the novice user's experience is described in sequential order of the approved protocol, followed by the activities of the expert user. The section ends with the evaluation criteria associated to the requested

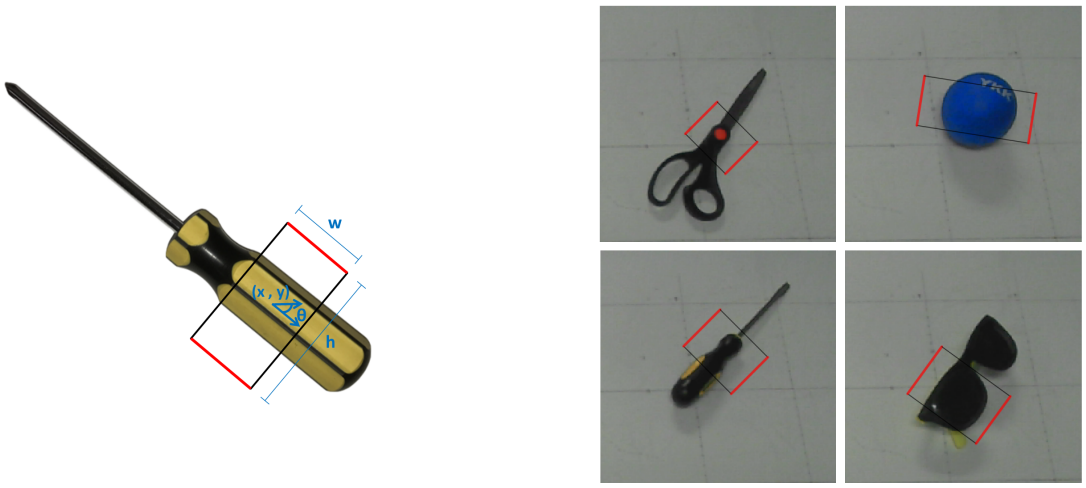

Figure 6.6: Left: The 5D grasp representation. Red lines correspond to parallel plates of the gripper. Black lines indicate opening distance of the gripper plates prior to grasping. Right: Examples of grasp outputs for several objects.

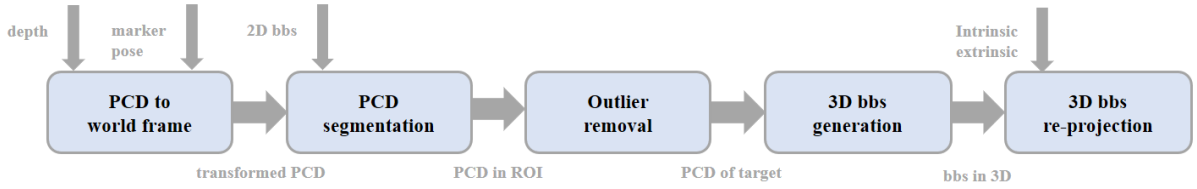

Figure 6.7: The 2D bounding box to 3D bounding box pipeline for grasping. It relies on the RGB-D data and known camera extrinsic parameters.

tasks.To evaluate the proposed pipeline, the system is validated through a series of experiment with human subjects.

6.4.1 Headset Interface Setup: All Users

During the investigation, each participant is required to wear the AR glasses and the TDS, both of which are headgear. To permit both to be worn at the same time and alleviate any weight imbalance issues, the META AR headset was attached to an adjustable headband with tension knobs for improved fitting to the subject. The TDS system was then placed over the headband and fit around the META AR headset. The two magnetic sensors of the TDS are adjusted so that one lies to the left of the participant's mouth and one to the right of the mouth, with both extended

out to lie around the lips. The relative pose between the META glasses and the TDS remain fixed throughout the experiments. Interfacing with the TDS requires a small disk-shaped permanent-type magnet tracer to be placed on the top surface of tongue close to its tip (around 1cm from the tip), as shown in Figure 6.4b. After dried out by a paper towel, a cyanoacrylic tissue adhesive glue is applied between magnet and tongue for attachment. attachment is temporary for this study, while the tongue piercing can be done at the same place for the long term usage.

6.4.2 Calibration of the TDS: All Users

For the TDS system to correctly recognize user tongue commands, each participant performs a calibration step with the TDS. The calibration attenuates possible EMF including the earths magnetic field and the META, in order to adopts to various head poses during experiments. During EMF cancelation calibration, 1000 magnetic sensor data are collected while the user keeps their tongue in a resting position and rotates their head. Since the magnetic sensor on the top of the head is only influenced by EMF, the EMF portion of the other 4 sensors can be cancelled using linear least square fitting and coordinate transformation. The subjects are simply asked to move/rotate their necks for several seconds while keeping the tongue stationary. A follow-up training period asks the user to provide training data for each command (up, down, left, right, and rest) 3 times in randomized order. Then, an RBF SVM model is trained. The overall calibration takes less than 5 minutes and re-calibration is required only when EMF changed or user switched. During operation, the SVM based algorithm performs classification using 10 past EMF cancelled data every 10 ms. If all the classification results are the same, the command is triggered and sent to the AR system.

6.4.3 Simple Manipulation Tasks: Novice Users

The overall pipeline involves interaction between multiple components including the TDS, META glasses, vision detection and robotic arm operations. The following tasks are designed to lead human subjects to get familiar with the system, and to achieve the *pick-and-place* task step-by-

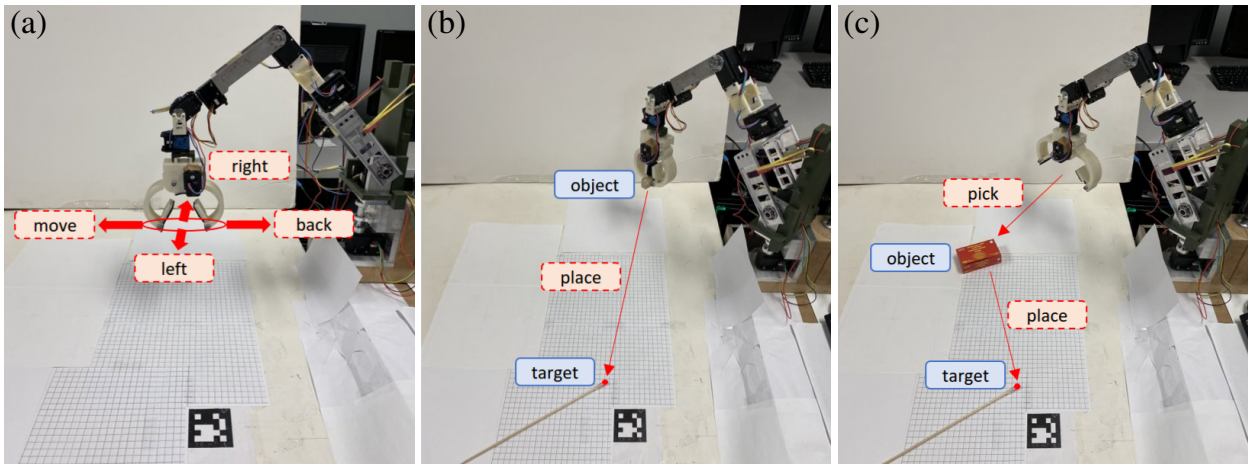

Figure 6.8: Illustration of three tasks involved in human subject experiments. (a) the command following task: subjects are asked to control the end-effector move toward to one of the four directions via the TDS and AR menu; (b) the placing task: subjects are asked to use the AR and the TDS to trigger the assistive robot, in order to place an object onto the target specified by the red dot; (c) the pick-n-place tasks: subjects are asked to operate the AR and the TDS to pick up an object and place to the target location.

step.

Task 0: Command Following

This warm-up task is designed to evaluate the performance of mapping the TDS commands to actual robotic arm movements through AR menu. As a warm-up task, participants learn the relationship between the TDS, the AR glasses and the robotic arm. During the warm-up task, participants are required to select from virtual buttons on the menu in AR glasses with tongue movements, in order to execute a corresponding robotic movement. Participants are instructed to move the robotic arm in four different directions: *forward*, *backward*, *left*, *right*, as shown in Figure 6.8a The instructions are provided randomly, and participants need to select buttons as quick as possible to move the end-effector accordingly. Each round includes 5 random commands. The response time and accuracy is recorded.

Task 1: Placing Task

This task evaluate a fundamental manipulation task: *placing task*. The placing task involves controlling the robotic griper to hold an object and move it to a desired location with the proposed

system. At the beginning of the *placing task* sub-session, a graspable object is provided by an instructor and held within two fingers of the gripper. Participants are required to operate the TDS and the AR glasses to trigger the *grasp* button to grab the provided object. After the object is held by the end-effector, participants are asked to place the object to a desired location. The desired location is randomly assigned by the instructor in the workspace within reachable range. The initial location to the desired location ranges from 30cm to 60cm, randomly set within reachable range of the manipulator. As shown in Figure. 6.8b, the desired location is assigned by the instructor with a wooden stick, where the tip of the stick is painted in red. The red dot indicates the desired location for placing. The instructor removes the stick after the assistive arm starts to move from the initial location. Placement success means the object is within a 1cm radius of the specified location. For the participants, the placing location is determined by looking at a calibrated red marker in the AR display, and overlapping the marker to the desired location in the real world. The robotic arm executes when the *place* button in the AR menu is activated through the TDS. The process is illustrated in Figure 6.8b. The success rate and speed are recorded.

Task 2: Pick-and-Place Task

The *Pick-and-Place* task aims to evaluate the effectiveness of the whole pipeline of the proposed system. With the vision algorithm for object/grasp detection, this task involves manipulation process to successfully pick up a randomly placed object in the workspace, and move it to a desired location. For each trial of the *pick-and-place task*, a graspable object is randomly placed in the workspace within reachable range. Participants are asked to operate the robot arm to grasp the object through the TDS and the AR glasses. By having the object in the view of the AR glasses, participants is able to execute the robotic arm to grasp the object by triggering the *grasp* button on the AR menu. Next, as in the *placing task*, participants are required to place the held object to a desired location randomly assigned by the instructor. Participants determine the location by overlapping the AR marker with the desired location, and trigger the *place* button through the TDS. The whole process is illustrated in Figure 6.8c. The success rate and speed are recorded. For *task 2*, a

task is counted as complete if the object is successfully picked up, and placed to a location within a 1cm radius of the specified location. The moving distance between initial end-effector location to the pick-up location is larger than 20cm; the moving distance between pickup location to desired location is larger than 20cm, both are randomly set within reachable range of the manipulator.

6.4.4 User Evaluation Study: Novice Users

Fatigue and Usability

All the 20 naive participants without previous experience are asked questions regarding to fatigue and usability. Both factors may impact the performance. The fatigue questions are related to involved body parts after all tasks are completed in the 3-hour session. And the usability questions are mainly for understanding difficulties of designed interfaces and tasks. The questions and corresponding results are shown in Table 6.2.

Cognitive Burden and Autonomy

The 20 naive participants are also asked questions regarding to cognitive burden and autonomy. Different from conventional interfaces, AR glasses enable *ego-centric perspective* with *summarized user intents*. The questions are centered around these two features. The summarized intents allow visual interpretation and therefore high level control; the ego-centric view prevents gaze disconnection on target and misalignment between user and system. Specifically, the ego-centric view enables users to see through the augmented menu, and determine a location for placing by simply *look at* a target. The questions and corresponding results are shown in Table 6.3.

6.4.5 Manipulation Tasks: Expert User

Besides essential activities such as *pick-n-place*, the proposed system is suitable for many common utilized manipulation in daily life. Performance of our system on more manipulation tasks are evaluated and summarized in Table 6.6. The results are compared with manual Cartesian control

by an expert user, for success rate and required time in different manipulation stages. The tested tasks are as follows:

Task 3: Open Drawer

A task to grasp the small knob of a drawer and pull in the perpendicular direction to drawer face in order to open it, then to place a target object into the drawer. The drawer dimensions were 14.6cm (depth) $\times$ 16.8cm (width) $\times$ 20.3cm (height) with three drawers of the same height. The knob is approximately 1.3cm in diameter. The *Open Drawer* task is separated into three stages: pulling out the drawer, reaching for the target object, and putting the object into the drawer.

Task 4: Salt Shaker

A task to grab a slat shaker from the side, move the shaker over the plate, and shake 3 times with the shaker facing downwards. The cylinder-shaped salt shaker for the experiment was 10.8cm in height and 4.9cm in diameter, while the plate was 22.6cm in diameter. The *Salt Shaker* task is split into two stages: picking up the salt shaker and moving it over the plate, then shaking the salt shaker and putting it back on the table. One failure case is due to incorrect visual estimation of the salt shaker location leading to a missed grasp. The overall processing time is again faster than Cartesian control in all stages.

Task 5: Unscrew Cap

Manipulation involving accurate translation and rotation to remove the cap from a bottle fixed to the table. The bottle's height was 18.6cm and the cap's diameter was 3.0cm. The *Unscrew Cap* task is a single-stage procedure.

Task 6: Towel Pick-up

Manipulation of a deformable and flat object requiring nail-like finger design of gripper for grasping. The square towel had a side-length of 34.5cm and thickness of 0.3cm. The *Towel Pick-up* task

is a single-stage procedure.

Task 7: Place Marker into a Pencil Box

A manipulation task to grab the marker from the top end then to place it into pencil box. The task requires orientation changes of gripper and obstacle avoidance while completing the task. The rounded pencil box had a diameter of 9cm. The *Place Marker into a Pencil Box* task is a single-stage procedure.

6.4.6 Performance Evaluation: All Users

Reaction Time (RT)(s)

represents the amount of time required to move the robotic end-effector through the TDS and META after receiving instructions in task 0. Final reaction time is reported as the mean value:

$$RT_{avg} = \frac{1}{m}\sum_{i}\frac{1}{n}\sum_{j} RT_{ij}. \tag{6.1}$$

where RT_{ij} is the time of the i_{th} trial for the j_{th} instruction. RT_{avg} is the time averaged over n instructions for each subject, and is averaged over m trials.

Success Rate (SR)(%)

represents the percentage of correct movements following the instructions in task 0. SR is determined as the ratio of sum of correct movements to the total number of instructions. Final success rate is averaged:

$$SR_{avg} = \frac{1}{m}\sum_{i}\frac{1}{n}\sum_{j} \mathbf{1}\{success_i\} \times 100\%. \tag{6.2}$$

where SR_{avg} is averaged over m trials, with each trials involves n instructions.

Task Completion Time (TCT)(s)

quantifies the required time to complete the pick and place movement in task 1 and task 2. The final task completion time is reported as the mean value, which is averaged over successful cases:

$$TCT_{avg} = \frac{\sum_i TCT_i \times \mathbf{1}\{success_i\}}{\sum_i \mathbf{1}\{success_i\}} \tag{6.3}$$

where TCT_i indicates the completion time for the i_{th} trial, $success_i$ indicates if the i_{th} trial is successful.

Task Completion Ratio (TCR)

quantifies ratio of the task (task 1 and task 2) that was completed by a participant depending on the threshold. The final ratio is shown as the averaged completed tasks over the total trial number m:

$$TCR_{avg} = \{\sum_i \mathbf{1}\{success_i\}\}/m \tag{6.4}$$

where $success_i$ indicates if the i_{th} trial is successful, and $m = 5, 10$ for task 1 and task 2, respectively.

6.4.7 Human Subjects

The system is evaluated by twenty recruited healthy/able-bodied subjects, aged 20 to 37 years old. All testing was approved by the Institutional Review Board of Georgia Institute of Technology under Protocol H19209. The experiment is limited to 3 hours and involves one session per subject, with three divided parts including preparation, training, and execution. For the consistency, all the experiments are held with the same equipments, and instructed by the same instructor, for all tasks. There were 20 novice subjects in total.

For each task, the subject was informed about the task and the outcomes to be recorded. For task 0, the outcomes recorded were the response time and success rate (in terms of how many movements matched the instructions). For task 1 and 2, they were the amount of time required to

accomplish the tasks, and the success rate (in term of 1cm radius threshold). The user was also told that, if the object was not picked up or was dropped during the task, it would be counted toward a failure case. After being informed of the task, the subjects were allowed to practice it with the ARS+TDS until they felt ready to officially perform the repeated trials. Once the subject finished the three tasks sets, they completed the suvery with questions related to fatigue, usability, cognitive burden and autonomy.

The expert user is a study member with some experience with the AR+TDS interface (approximately 30 hours on AR+TDS). The expert user followed the same routine as the novice users including preparation, training, and execution. Except that for the expert user, only the pick-and-place task with 10 objects were executed. To prevent from re-training the TDS model, 10 objects are executed in one sitting session.

For each of the session, the same instructor helped the user to walk through the preparation, training and execution. During the preparation, the instructor was in charge of explaining the experimental protocols and details of each tasks. After the subject and the instructor signed the consent form, the instructor helped to attach the magnet, the TDS and the AR glasses for the subject. During the training process, the instructor operated the main laptop (with TDS software) to guide the user to calibrate and train the SVM model for the TDS. The details of the TDS training process is in Sec. 6.4.2. The TDS signals (triggered by the subject) were fed to the main laptop for the instructor to monitor on the screen. This allowed the instructor to keep track of the status of the TDS (properly connected) and the triggered signals (properly transmitted). After the training of the TDS, the instructor assisted the subject for tasks. For each of the task, the instructor recorded the related data (success rate/failure reason/time/error). The instructor no longer monitored the main laptop after training. Note that there was a second monitor mirroring the menu seen by the subject for the experimental setting. This allows the instructor to visually confirm whether the virtual button is properly triggered by the subject. Also, this could be used for confirming whether the TDS triggering event and the virtual button triggering event are consistent. We never observed the inconsistency between the TDS and the virtual button, therefore we mainly used the first monitor

and this second monitor was not used during the session. For pick-and-place task, the instructor was also in charge of placing the object in the workspace as described in Sec. 6.4.3. Before the session ends, the instructor was in charge of making sure the attached magnet was properly removed before the subject was allowed to leave. Usually it took only 10 to 15 minutes. Overall, the role of the instructor is to explain the protocol, attach devices, monitor the signals, record related data, resolve technical issues, and had document properly signed, through each session.

6.5 Results and Discussion

This section describes the outcomes associated to all of the tests performed. Where the tests are similar the outcomes are compared. Likewise, published literature with comparable tests and outcome statistics are compared. After presenting the results, each section also includes a brief discussion specialized to the tests performed and the available statistics.

6.5.1 Performance Results: Novice Users

The simple interface and manipulation tasks completed by the novice subjects incrementally introduced the subjects to the assitive robot interface and its execution characteristics.

Task 0: Motion Commands

Being the first time warm-up use of the assistive robot interface, *Task 0* provides the worst case timing and human-induced error rates for the system. These statistics are provided in Table 6.1. The average *Reaction Time* across all participants was 6.36s. The success rate of 95.6% indicates an error rate of 4.4%. More than half of the participants (11 of 20) achieve a perfect success rate, which shows that the TDS is a relatively natural interface to learn for selecting commands from a virtual AR menu. The remaining 9 out of 20 participants triggered incorrect robotic movements in some of the cases, with this population fitting one of two categories. The first error category was due to mistakenly triggering the same command for the second time (3 subjects). Per Sec. 6.3, the system was designed to provide visual feedback of the robot's activity state as well as a one second

Table 6.1: Experiments on Three Tasks with 20 Healthy Human Subjects

	task 0		task 1		task 2			
participant ID	SR	reaction time	TCR	task complete time	TCR (pick)	task complete time (pick)	TCR (place)	task complete time (place)
01	100	6.00±0.55	4/5	18.20±5.59	9/10	24.67±5.98	7/10	15.56±3.91
02	100	6.40±0.60	2/5	19.20±3.27	9/10	26.78±9.30	7/10	17.00±6.73
03	84	6.60±0.45	4/5	16.00±3.32	9/10	18.78±1.20	7/10	16.88±1.96
04	100	5.88±0.41	3/5	14.60±5.41	10/10	19.80±1.23	9/10	14.90±1.29
05	96	6.76±1.18	5/5	22.60±1.82	10/10	24.60±6.04	8/10	16.60±1.17
06	100	6.44±0.48	5/5	16.00±3.32	10/10	20.20±3.33	9/10	18.60±4.45
07	68	7.56±0.93	4/5	19.80±3.19	8/10	23.00±7.91	7/10	18.00±6.92
08	96	5.88±0.18	4/5	26.40±7.77	9/10	31.78±2.86	8/10	14.89±1.45
09	100	8.84±1.68	3/5	19.40±2.41	8/10	28.00±10.30	7/10	17.38±3.34
10	100	6.36±0.48	4/5	20.20±2.59	10/10	28.70±3.06	9/10	15.50±1.58
11	100	6.35±0.86	5/5	21.80±9.98	9/10	34.88±4.67	8/10	15.38±1.60
12	92	6.44±0.86	3/5	18.60±8.26	8/10	21.14±5.11	6/10	14.86±0.69
13	100	5.76±0.83	5/5	19.20±3.94	9/10	25.67±2.00	8/10	17.88±6.96
14	92	5.44±1.27	5/5	23.40±2.88	6/10	18.33±2.25	6/10	15.50±3.45
15	96	7.64±1.56	4/5	30.00±8.25	9/10	23.33±6.52	8/10	20.50±8.47
16	92	5.52±0.58	5/5	24.40±4.16	7/10	22.00±6.24	7/10	13.71±5.41
17	100	5.48±0.18	4/5	16.80±2.39	10/10	17.20±1.99	9/10	15.40±3.06
18	100	6.08±0.63	5/5	16.60±2.07	10/10	17.20±1.23	9/10	15.50±2.51
19	100	6.12±0.72	3/5	30.40±4.51	9/10	18.67±0.87	8/10	16.33±1.12
20	96	5.56±0.38	5/5	28.80±5.59	9/10	20.11±2.03	9/10	14.78±1.92
average	95.6	6.36±1.12	4.10/5	21.04±6.19	8.90/10	23.19±6.67	7.80/10	16.26±4.08

* SR: Success Rate; TCR: Task Completion Ratio.

motion pause at the conclusion of a movement before accepting new commands. The 3 participants failed to move their tongue back to the *rest* position during this time period in preparation for the next command. Additional practice will reduce this form of error, as would incorporating interface changes such as waiting for the *rest* state to accept new commands. The second error category is due to the sensitivity of the TDS to the tongue location. The TDS signal interpretation misinterprets the commands due to difficulty in making distinguishable tongue poses. Observationally, the main confusion was with *up* and *down* commands, stemming from the left/right positioning of the TDS sensors and the lower signal differential associated to vertical tongue movements. On the user side, improvement may be possible with better calibration or TDS sensor positioning, or through practice. On the interface side, limiting the interface to tongue poses with low confusion rates would prevent missclassification.

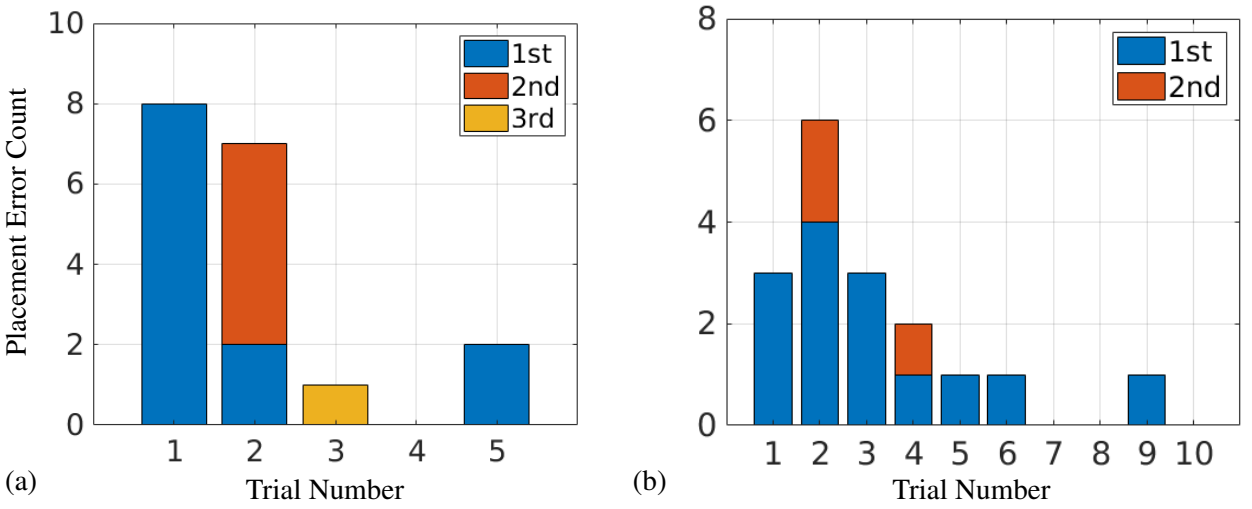

Figure 6.9: Distribution of placement errors for the *pick-and-place* task for novice users for (a) Task 1 and (b) Task 2. The color coding is the mistake number since some participants committed more than one mistake.

Task 1: Grab and Place

Task 1 involved two commands, one to grab and one to place, with one chance each to do so correctly. All of the commands were successfully executed by 19 of the subjects, with 1 subject committing an error. The one participant mistakenly triggered *place* before aiming at the target, which led to a motion planning failure (unreachable target location). This gives a command triggering error rate of 0.5%, which is 8.8 times lower than the error rate from *Task 0*. The average *Task Completion Time* for *placing task* is 21.04s. The task completion rate reflects what percentage of objects were placed within the 1cm threshold radius for being classified as a success. The failure case breakdown indicates 1% of attempts leading to failure due to human input error and 17% due to placement error. Based on the *Expert*'s perfect placement (see Table 6.4), the placement error is predominantly a function of user pointing error with the AR device. The *place* error stacked histogram in Figure 6.9(a) shows the error counts versus the trial number across all subjects. The 18 errors were due to 12 users, with the remaining 8 having perfect performance (also quantified in Task 1 TCR column of Table 6.1). Of these 12 users, 4 had two mistakes, and 1 had three mistakes. Roughly 40% of subjects were able to use the TDS+AR without error, 35% committed one error,

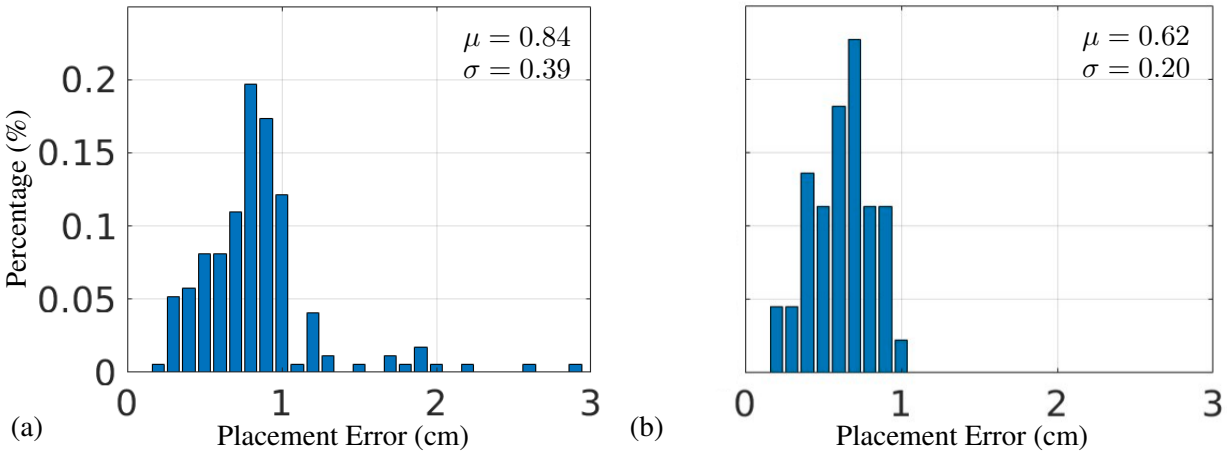

Figure 6.10: Distribution of placement errors for the *pick-and-place* task for (a) novice users and (b) the expert user.

and 25% committed two or more errors. With further practice, the placement error should improve.

Task 2: Pick and Place

Task 2 also required two sequential commands, both involving targetting an object or location with the AR interface before selecting the action to execute. The *Task Completion Time* for each subject reflects these two steps and is split into *pick-up* and *place* sub-task timings. If the object is dropped during placement, it does not contribute to the mean and variance of the *Task Completion Time* column for the *place* sub-task.

Regarding the TDS input modality, the first command of the sequence was completed by all subjects. The second command of the sequence was incorrectly triggered early by two subjects, for a command triggering error rate of 1.0% (4.4 times less than that of *Task 0*). The failures associated to the *pick-up* sub-task are due to visual processing failures and agree with the 11% grasp failure rate of the underlying algorithm [97]. The *place* sub-task failures split into command triggering human-error failures, object grasp failures, and large positioning error. The object grasp failures are indicative of poor grasps not robust to robot movement, of which there are three. It is best attributed to a failure in the robot to either predict the correct grasp or to correctly execute the planned grasp. For positioning error, there were 17 cases. Overall, the task outcome error

percentages are 1% due to human input error, 12.5% due to robot-error, and 8.5% due to human pointing error. Figure 6.10(a) provides the distribution of placement error distances for *Novice* users. Increasing the threshold to 1.50 cm leads to a 5% human pointing error percentage relative to all task instances (10 out of 200). If the placement radius is further enlarged to 1 inch (2.54 cm), then the placement error reduces to a 1% human pointing error rate (2 out of 200). Figure 6.10(b) provides the same distribution for the *Expert* user. All of the outcomes are within 1 cm. Compared to the expert user, the novice users have larger placement error in average (0.84 v.s. 0.62), and produce additional large outlier placement errors.

Per Figure 6.9(b), which is a stacked histogram of error counts versus trial number across all subjects, 14 subjects committed errors, with 3 of them committing two errors. The other 6 subjects performed perfectly (as quantified in the Task 2 TCR (place) column of Table 6.1). Here 30% of subjects used the TDS+AR perfectly, 55% had one placement error, and 15% had two errors. The distribution of errors is biased towards lower numbered runs, much like in Figure 6.9(a), which suggests that some accomodation is needed when performing new tasks or action sequences, however the low counts for higher numbered runs indicates that some errors may not be associated to the user, but possibly to the autonomous sub-system of the robot. A combination of more practice and better algorithms would improve these success rates.

Discussion

The *Novice* user's human subjects tests indicate that the AR+TDS is an easy to learn interface for high-level, hands-free control of an assistive robotic arm. The predominant sources of error for grasping are due to the underlying autonomy stack (around 23.5% when considering grasping errors across the *pick* and *place* sub-tasks), while the prevailing sources of object placement errors are due to imprecise pointing with the AR headset (8.5%). The former requires improvements to the underlying visual processing algorithms, while the latter is easily addressed by continued practice. Nevertheless, for less strict placement, this percentage can drop to 5% or 1% of all attempts. Given that not all object placement needs to be precise to 1 cm, *Novice* users should

Table 6.2: Fatigue and Usability Evaluation

Theme 1: Fatigue Questions		
	Question	Yes / No
Q1	During the all the experiments, was your tongue tired?	10/10
Q2	During the all the experiments, was your jaw tired?	5/15
Q3	During the all the experiments, was your neck tired?	3/17
Q4	During the all the experiments, were your shoulders tired?	1/19
Theme 2: Usability Questions		
	Question (from 1:very difficult to 5:very easy)	mean$\pm$std
Q5	How difficult is it to use the Tongue-Drive System to control the robotic arm?	3.20 ± 0.93
Q6	How difficult is it to use the Tongue-Drive System to trigger buttons in META glasses?	3.40 ± 1.02
Q7	How difficult is it to use the Tongue-Drive System and META to place the target?	4.00 ± 0.84
Q8	How difficult is it to use the Tongue-Drive System and META to pick the object?	3.85 ± 0.91
Q9	How difficult is it to use the Tongue-Drive System and META to complete the pick-n-place task?	3.55 ± 0.97

learn to execute *pick-and-place* tasks without significant training demands.

6.5.2 Interface Results: Fatigue and Usability

The questions asked of the users and their responses are given in Tables 6.2 and 6.3. The intent behind the questions is to gain understanding on the two interface mechanisms, the TDS for input and the AR for visual feedback, menu visualization, and shared perspective.

Fatigue

The fatigue questions revolved around the main muscle groups used during the experiment, from the shoulder to the tongue. They were yes/no questions. Initially, the hypothesis was that the weight of the headset would be the dominant source of fatigue. The AR headset and the TDS are mounted to a specialized headstrap for distributing the weight over the top of the head. Without it, the AR headset uses a compression fit around the side of the head plus a support point on the nose to stay in place. The META 1 AR headset is front-heavy and can cause discomfort around the nose. The headstrap corrects for some of this weight imbalance. However, the survey results in Table 6.2 indicate that this is not the case. There were more positive fatigue responses for the tongue and jaw than for the neck and shoulder. Looking at the three responses for neck, two of the subjects also

experienced fatigue for the jaw and tongue, and the rest one also experienced fatigue for the tongue. The single response of fatigue at the shoulder also experienced fatigue at the jaw. These subjects reflect a group for which the AR and the TDS will require some training and accomodation time. The remaining responses had 5 subjects who rated the tongue as the only source of fatigue, and 2 subjects who rated the tongue and jaw as a source of fatigue. More practice with the TDS would alleviate those values for this group. The remaining 9 subjects experienced no fatigue whatsoever. Overall, fatigue ratings increase when moving from the larger shoulder muscles to the smaller tongue muscles. Almost half of the participants (45%) feel no fatigue. The overall setup of an AR-based headset with a hands-free selection interface appears to be a reasonable interface design option.

The fatigue outcomes correspond to 20% of novice users requiring acclimation to the TDS and the AR, 35% to the TDS alone, and 45% requiring no acclimation. More recent AR headset designs have improved designs that better distribute the weight around the head and support the headset from above with head straps rather than via compressive straps and a nose bridge. Furthermore, they tend to be lighter. Some of the fatigue problems associated with the shoulder and neck should go away with more modern AR gear, as well as with future designs. These headset improvements would address the acclimation needs of the first group (20% of novice subjects).

Throughout the experiments, the tongue is the most engaged muscle with the jaw a close second. Movements of the tongue also often involves jaw movement to open up space for the tongue to reach. The *up* and *down* commands require more exagerated movements since they are harder for the TDS to differentiate. Vertical movement is orthogonal to the axis of most sensitivity for the TDS sensors. An improved TDS design, more sensitive to movements and possibly requiring less exagerated movements, would improve fatigue responses– and acclimation–to the TDS. Improving the sensitivity of the design to vertical movements by adding more sensors, or exploiting the horizontal motion sensitivity to create more commands in the horizontal plane, are both options to consider in future iterations of the TDS system. These modifications could improve the TDS experience and acclimation for 55% of novice subjects.

Table 6.3: Cognitive Burden and Autonomy Evaluation

Theme 3: Summarizing User Intents			
	Question	Yes / No	mean±std
Q10	How much do you think summarizing user intents in the menu simplifies the task difficulties?	–	4.20±0.81
Q11	Do you think summarizing user intents in the menu help to reduce your cognitive burden?	16/4	–
Q12	Do you think summarizing user intents in the menu helps to improve the autonomy when completing the tasks?	18/2	–
Theme 4: Sharing Ego-centric Views			
	Question	Yes / No	mean±std
Q13	How much do you think "red dot" in augmented reality glasses help to locate the placing location?	–	4.50±0.59
Q14	How much do you think ego-centric view helps to understand the relative locations between you and object?	–	4.25±0.70
Q15	Do you think the ego-centric view helps to reduce the cognitive burden?	18/2	–
Q16	Do you think the ego-centric view helps to improve the autonomy when completing the tasks?	19/1	–

Usability

The usability questions permitted answer on a 5-point Likert scale, with 1 (very difficult) and 5 (very easy). The average difficulties of using the TDS to trigger virtual buttons, and to control the robotic arm are 3.20 and 3.40, respectively. For Q5, since this relates to a relatively easy warm-up task (task 0), we found only subjects who achieved 100% success rate gave positive or highly positive. The majority rated neutral (9 subjects) or higher (7 subjects). For four of the subjects who rated 2, there are two subjects achieved 100% success rates. One of these two subjects reported experiencing fatigue in tongue (Q1) and jaw (Q2). The other one also voted 2 in Q6 and thus may have non-positive experience in triggering buttons to controlling the robotic arm. For Q6, the majority rated neutral (4 subjects) or higher (11 subjects). Of five subjects voting 2, two subjects also reported fatigue in tongue (Q1) and jaw (Q2), one achieved the worst overall success rate. One used the longest completion time in Task 1. Thus they may have negative perception or experience in working with TDS+AR system while triggering buttons.

The average difficulties of placing and picking are 4.00 and 3.85, respectively. By overlapping the virtual marker with desired location, participants easily determine location for placing and the

proposed system autonomously plans and executes the action. For the question related to placing with TDS+AR system, only two subjects gave negative responses (no highly negative). And they were the two subjects with the worst overall placing success rates. Besides, the only subject who voted neutral is the third-to-last person. The rest of 17 subjects who had better success rates in placing tasks had positive or highly positive responses. For the question related to picking, only two subjects gave negative responses. And they all had performance below average (one of the two had the worst picking success rate). For the three subjects gave neutral responses, only one is slightly above average and the rest are below average success rate. For the question related to pick-and-place task, 17 subjects voted neutral (4 subjects) or higher (11 subjects) with the average score to be 3.55. Only three subjects gave negative response (no very negative responses). Also, two of these three subjects had the worst pick-and-place success rates.

Cognitive Burden

Questions related to the use of the AR menuing system and its perceived contribution towards task simplification are shown in Table 6.3, together with the response statistics. For the ordinal question, Q10, the values range from 1 (highly complicates the tasks) to 5 (highly simplifies the tasks). The high average score (4.20) and low standard deviation (0.81) indicate that most subjects (17 of 20) perceive that summarizing user intents via the menu simplifies the task difficulties (Q11). Two subjects gave a neutral response and one subject gave a negative response (no very negative responses).

Continuing, the majority of subjects believe that summarizing user intents via the menu system contributes to reduced cognitive burden and facilitates autonomous operation. The 4 No responses to Q11 also includes the 2 subjects who responded No to Q12. Their performance outcomes (overall success rate vs. cumulative time averages for Tasks 1 and 2) are plotted in Figure 6.11. The horizontal and vertical lines are located at the average values and partition the space into quadrants. Three of the subjects score near to or below average in success rate and also below average in completion times, i.e., they are fast but less accurate. Their placement suggests that

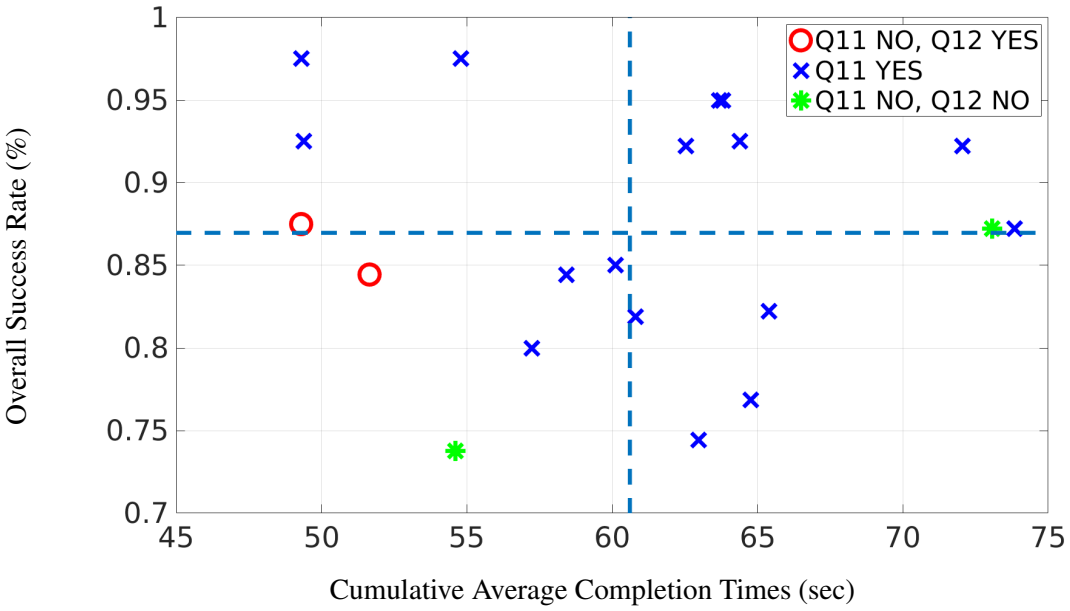

Figure 6.11: Performance outcomes for sub-populations based on their responses to Q11 and Q12. Both axes aggregate the Task 1 and Task 2 outcomes.

the subjects operate sufficiently fast that they may not be sensitive to any potential benefit arising from interfaces enhanced by autonomous capabilities. At the opposite extreme, the one subject with a long completion time and average success rate may also not be sensitive to any benefit due to high task completion times. This subject also experienced fatigue at the neck, jaw, and tongue. The negative perception may be a function of their overall difficulty working with the TDS+AR system, which would improve with practice.

Shared Autonomy

The questions related to autonomous robot operation revolved around the shared ego-centric view afforded by the AR headset, as provided in Table 6.3. While other interfaces such as monitors or tablets could potentially summarize intents as well, the shared ego-centric view is unique to AR glasses. It permits pointing to and selecting objects or locations through head pose fixation. It also permits the presentation of action options without needing to look away from the task at hand. The scores with ordinal answer options were on a 5-point Likert scale with 1 (highly unhelpful) to 5 (highly helpful).

Table 6.4: Comparisons of Cartesian controlled (left number) and TDS+AR controlled (right number) methods on pick-n-place

Cartesian / TDS+AR	picked up	pickup time (s)	placed	place time (s)	# commands
stapler	5 / 4	69.5±7.7 / 14.1±1.5	5 / 4	77.2±8.9 / 16.9±1.0	14.6 / 2
spoon	5 / 5	63.4±8.7 / 13.7±1.1	5 / 5	77.7±19.1 / 17.5±0.4	10.4 / 2
banana	5 / 4	67.0±20.7 / 16.2±0.6	5 / 4	64.4±15.4 / 15.7±0.3	11.2 / 2
screw driver	5 / 5	68.0±7.7 / 14.4±1.9	5 / 5	63.6±13.6 / 17.4±1.0	11.4 / 2
bowl	5 / 5	53.6±2.1 / 16.6±0.9	5 / 5	84.8±26.6 / 15.6±0.4	9.6 / 2
ball	5 / 3	66.6±9.9 / 16.6±3.2	5 / 3	87.3±16.3 / 17.8±1.1	11.6 / 2
sunglasses	5 / 5	62.1±4.8 / 16.3±0.4	5 / 5	69.5±16.3 / 15.6±0.6	10.0 / 2
pliers	5 / 3	74.9±6.6 / 16.7±0.6	4 / 3	92.4±24.0 / 15.0±0.8	6.5 / 2
scissor	5 / 5	63.8±7.0 / 15.9±0.5	4 / 5	69.4±10.0 / 15.8±1.3	4.2 / 2
tape	5 / 5	60.5±7.9 / 16.1±1.1	5 / 5	72.0±19.2 / 15.9±1.3	4.6 / 2
average	5.0 / 4.4	65.3±9.4 / 15.6±1.6	4.8 / 4.4	76.2±17.2 / 16.3±1.2	9.4 / 2

Both Q13 and Q14 have high average scores and low standard deviations, which means that most of the responses were positive, in the sense that both the ”red dot” and the shared view were perceived to be positive features of the shared-autonomy system. Four subjects rated these properties of the shared autonomy system as neutral (there were no negative responses). One of the four also responded No to Q11 and Q15, Thus, this subject generally had non-positive views of the system from the Cognitive Burden and Shared Autonomy perspectives. Of the other three subjects who rated as neutral in Q14, one was with the longest completion time but only average success rate, one had low success rate (the second worst), and the third one had success rate below average while completion times above average. Generally, their performance (lower success rate, longer completion time) suggests non-positive experience in gaining benefit from shared autonomy. Outside of these four, the general consensus is that the shared view and its ”red dot” visual is a helpful feature of the shared-autonomy assistive manipulator.

6.5.3 Results: Expert User

The expert user performed two sets of experiments that serve to demonstrate the upper bound of performance for the system as currently designed, provide comparison to other existing hands-free interfaces, and demonstrate the future value of the shared-autonomy pipeline. The first experiment is the pick-and-place task that serves as the canonical testing task, and the second consists of

executing more diverse manipulation tasks as described in §6.4.5.

Table 6.5: Comparisons with Existing Hands-Free Interface Research

pickup only / pick-n-place	experience	# of subject	success rate (%)	time (s)	# of objects	# of trials	modality	autonomy
[151] w/Actuator	Expert	1	50 / –	70.1 / –	1	10	tongue	basic
[151] w/Cartesian	Expert	1	80 / –	71.3 / –	1	10	tongue	medium
[152]	Novice	1	– / 80	– / 65	1	5	tongue	medium
[152]	Expert	1	– / 100	– / 47	1	5	tongue	medium
[168]	Novice	5	82 / –	92 / –	3	5	sEMG	full
[169]	Expert	1	92 / –	14.2*/ –	1	50	eye	full
AR+TDS	Novice	20	89 / 78	23.2 / 39.5	1	10	tongue	full
AR+TDS	Expert	1	88 / 88	15.6 / 31.9	10	5	tongue	full

* path planning + execution time reported (excluding user observation time)

Low-level Control Comparisons

Due to the ease and more flexible time constraints associated with the expert user, this subject was tasked to pick up a variety of commonly seen objects (10 objects) all of which are, in principle, capable of being grasped by the robot's autonomous sub-system [97]. For each object there were 5 *pick-and-place* trials for a total of 50 tests. The expert was not given a second opportunity to pick up an object if the robot arm failed. The semi-autonomous AR+TDS and manual Cartesian control approaches were implemented with the same end-effector speed. In the latter, the user controls the manipulator end-effector with 9 commands via a keyboard interface (rotation, open and close end-effector, and 6-DOF movement). As such, manual Cartesian control should provide a lower bound on the best achievable performance for the manipulator used (a 7 DoF arm) and for any selection based mechanism with similar choices since other interfaces will be slower or have a lower throughput rate. Recorded metrics include *success rate*, average *task completion time*, and the *number of issued commands*. Table 6.4 provides the outcomes for the five experiments per object.

The shared-autonomy approach that summarizes robot affordances and automates task execution reduces the number of commands issued by the user (by factor of 4.7). The completion speed is 4.4 times faster then manually controlling the end-effector and has a lower standard deviation.

These reduction in commands and increased speed is a function of the simpler interface associated to high-level task commands, which should lead to a reduced cognitive burden on the user. The current limitation to the shared-autonomy approach is in the autonomy algorithms. Due to limitations in the grasp recognition algorithm [97], the success rate degrades (88% versus 96%). Considering that the probability of failure is 12% and assuming that repeated attempts have independent failure probabilities, then a second attempt would have a 1.44% failure rate and a third attempt would have a 0.17% failure rate, while taking on average 47.5s and 634.1s respectively. In either case, the time to pick and then place is still faster than the Cartesian interface (141.5s on average). One benefit of software-based autonomy is that improvements to the grasp recognition algorithm would improve the performance at little to no cost. For example, these grasp recognition methods in [142, 170] have a 95% and higher grasp recognition rate.

Comparison with Other Hands-Free Methods

Comparison to published research in Table 6.5 provides statistics for two commonly seen manipulation tasks: pickup and pick-and-place. For improved context the table reports the number of objects tested and trials per object, as well as other properties of the experiments. Three autonomy levels are categorized from low to high: *basic* directly controls manipulator joints; *medium* controls the end-effector; and *full* only requires users to select from candidate intents. The experiment in [171] is excluded due to an incompatible test scenario. The presented outcomes are sorted according to the autonomy levels, with the experience level being the secondary sort factor.

The 95% confidence intervals for the expert and novice success rates of the AR+TDS system are 88 ± 10.5 and 89 ± 4.69, respectively, for the *pick* sub-task, and 88 ± 10.5 and 78 ± 4.43, respectively, for the full *pick-and-place* task. Looking first at the *Novice* outcomes for the *pick* sub-task, there are no other methods with comparable *Novice* performance (e.g., lying above the lower interval limit of 84.31%). Expanding to include *Expert* outcome, the study results of [169] lie within the interval meaning that some *Novice* users of the AR+TDS can achieve the equivalent of *Expert* performance for the other interfaces. As mentioned earlier, the main limitation lies in

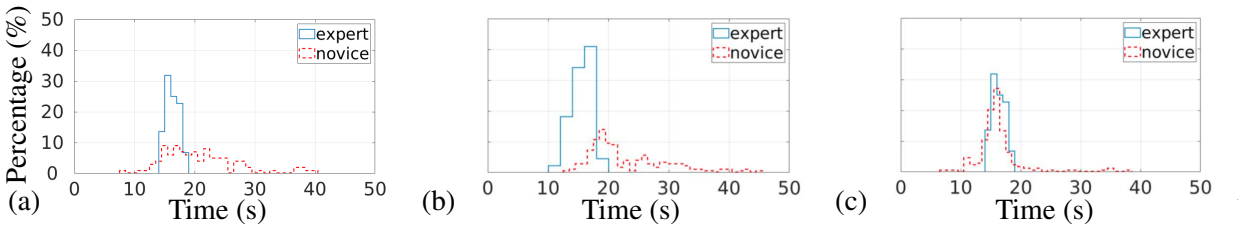

Figure 6.12: Time distributions for the *place* and the *pick-and-place* task for novice and expert users: (a) Distribution of Task 1 place time; (a) Distribution of Task 2 pick time; (b) distribution of Task 2 place time. The data for the *Expert* is from only the *pick-and-place* task data in Table 6.4.

the grasp recognition algorithm, which means that the success rate can only increase in future iterations. The *Expert* user of the AR+TDS has comparable performance to the *Novice* users, but comparison value is limited due to the more diverse objects tested by the *Expert* with more variable success rates.

It is more valuable to look at the difference in *Time to Completion* for the *pick* and *place* actions. The distribution of completion times, Figure 6.12, provides more information on what is happening. The data for the *Expert* user is plotted with solid blue lines and for the *Novice* users with dashed red lines. The first time that the *Novice* users engage in a *pick* operation or a *place* operation, the time distribution is spread out. In contrast, the *Expert* distribution has lower variance and exhibits a single mode. The second time the *Novice* users perform a *place* operation, the distribution becomes more peaked and overlaps the *Expert*'s distribution, though there are a few long duration outliers. These results suggest two findings. The first is that learning to command a *place* maneuver comparable to an *Expert* happens quickly, though it will still take some practice to do so with *Expert* precision, per Figure 6.10(a). The second is that *pick* somehow differentiates itself from *place* even though the actual TDS input sequence is nearly the same (see Figure 6.2). Since the repeated *pick-and-place* tasks interleave picking and placing, the difference in timing cannot be due to differences in experience. Further study is needed to understand why the *pick* task differs from the *place* task in term of operational time and skill acquisition. It might be related to the robot arms grasping error rate and internal attempts by the users to identify what point on the object to trigger the grasping routine.

Continuing to explore the timing, [169] reported lower times. However, their reported duration

only covers path planning (0.42s) and robot execution time (13.80s) whereas we include the whole duration from user observation to robot execution. The average vision process time of our system is 0.12s, and the manipulator path planning time is less then 0.10s. The execution time of each trial depends on the target locations (ranging from 11s to 12s), thus a reasonable upper bound on the comparable time is 13s, which is similar to the 14.2s reported. Examining the remainder, an approximate value for the average *Novice* interface time is 10s, while for an *Expert* it is under 3s on average. In general user observation and command time (to locate the target and trigger the TDS) will vary based on the relative geometry of the task. One additional difference is that [169] places the same object (a bottle) in the same location for all 50 trials. The AR+TDS system was tested with the objects in random locations.

Moving to the full *pick-and-place* task, the only comparable experiments are the two from [152] which consist of moving a bottle with a commercial robotic arm by sending *medium* level commands via a tongue interface. *Novice* success rates are comparable but with the ARS+TDS achieving a 39% lower completion time, and most likely requiring significantly less commands based on the different autonomy levels (the results in Table 6.4 indicate 4x less commands given). With regards to timing, the comparable *Expert* level outcomes have a 32% lower completion time, indicating that the completion time differential persists as users gain experience with each system. The AR+TDS success rate is lower. However, it is a function of the greater diversity of objects tested. Six of the ten objects had a perfect *pick-and-place* success rate (spoon, screw driver, bowl, sunglasses, scissors, and tape). As noted earlier, the success rate is a function of the underlying manipulation algorithms rather than being user error, and should improve as the system code is upgraded. Meanwhile, the speed-up affords an additional attempt for the *pick* sub-task while remaining competitive with [152] (for both the *Novice* and *Expert* levels).

Additional Manipulation Tasks

The *Expert* performance for the additional tasks when using the AR+TDS shared-autonomy system is summarized in Table 6.6. The results are compared with manual Cartesian control for success

Table 6.6: Cartesian controlled (left number) and TDS+AR controlled (right number) Manipulation Tasks

Cartesian / TDS+AR	success	stage 1 time (s)	stage 2 time (s)	stage 3 time (s)	# of commands
Open Drawer	10 / 7	40.2±7.2 / 13.6±0.7	58.7±8.8 / 25.1±1.4	82.6±19.5 / 16.3±0.7	21.9 / 2
Salt Shaker	10 / 9	134.6±25.5 / 21.7±1.7	51.4±9.5 / 25.2±2.6	– / –	21.2 / 2
Unscrew Cap	10 / 7	45.0±6.7 / 11.4±0.4	– / –	– / –	8.0 / 1
Towel Pick-up	10 / 10	66.3±6.0 / 13.0±0.9	– / –	– / –	9.0 / 1
Marker into Box	10 / 10	112.3±9.1 / 34.7±0.6	– / –	– / –	14.9 / 2
average	10 / 8.6	– / –	– / –	– / –	15.0 / 1.6

rate and required time in different manipulation stages. The time improvement for using the shared-autonomy assistive manipulation system varied from 57.2% to 83.9% faster than manual Cartesian control for the sub-tasks. These time improvements lead to a 69.7% faster time to completion for the complete *Open Draw* task, and a 74.8% faster time to completion for the *Salt Shaker* task. Overall, these values indicate task completion rates that are 3-4 times faster relative to Cartesian control. Importantly, the number of commands issues for the AR+TDS system is in the low single digits while manual Cartesian control (equivalent to *medium* level autonomy) requires nearly an order of magnitude more commands.

These benefits currently involve a trade-off in reduced success rate, with the AR+TDS being 86% successful and Cartesian control being 100% successful. The failure modes are discussed here. Two failures cases for the *Open Drawer* task were due to a missed grasp of the drawer knob, with the remaining failure due to the end-effector losing grip of the knob during pulling. Essentially, the drawer is the primary failure mode, while the grasping and placement of the object into the drawer was not. For the *Salt Shaker* task, the one failure case is due to incorrect visual estimation of the salt shaker location leading to a missed grasp. The failure cases in *Unscew Cap* are due to the precision grasp requirements for parallel-type grippers. The gripper must align the gripper wrist rotation axis with the cap normal axis and be centered at the cap. In real-world circumstances, repeat attempts would be permissible and, with high probability, would have a task completion time lower than Cartesian control. Nevertheless, future work will establish improved

manipulation primitives and visual processing methods for these tasks.

Overall the tests demonstrate the versatility of the AR+TDS guided manipulation framework on a range of tasks compatible with parallel-type grippers. The 86% success rate for common ADLs is close to that of the *Pick-and-Place* tasks, while the time savings and commands issued are also consistent. These outcomes suggest that shared-autonomy approaches to guided manipulation, using advanced computer vision algorithms combined with manipulation primitives, are effective strategies to pursue for hands-free robot assistance. Having an object and task adaptive menuing system would preserve the low command rates associated to executing the tasks.

6.6 Conclusion

We presented a collaborative human-robot framework based on a shared-autonomy paradigm for persons with high-level paralysis to guide manipulation tasks. The assistive system provides enhanced user autonomy by integrating vision algorithms with augmented reality (AR) and the tongue-drive system (TDS). As a human-in-the-loop system, the shared perspective and menuing system simplifies the control of manipulation tasks by interpreting the ego-centric view of AR, summarizing action intent by vision algorithms, and triggering execution via a menuing system using the TDS. The proposed system was validated by 20 able-bodied human subjects with no prior experience in AR and the TDS. Our system outperformed published state-of-the-art results at different autonomy levels in terms of trade-off in success rate and operation speed, while having a relatively small gap between novice and expert users. Evaluations regarding to autonomy and cognitive burden also indicate the benefit of the AR in shared view and summarized intents, as well as augmented vision effect for the assistive system. Additional experiments include common manipulation tasks in ADLs. They illustrate the effectiveness of our system through analysis of execution time and number of commands issued. Compared to Cartesian control, the system exhibited improvemed completion times through the simplification of complicated manipulation tasks via manipulation primitives. Future work will address the autonomous performance of the manipulation algorithms, and the TDS command miss rate for vertical tongque movements.

CHAPTER 7

CONCLUSION

This thesis investigates the robotic affordance understanding for application of vision-based manipulation. A series of improvements covering from training/inferencing efficiency to generalizability of learned model are achieved step-by-step. The overall efforts improve the applicability of affordance understanding in manipulation tasks, and bridge the research gap between vision benchmarks and real-world robotic missions. Multiple research gaps of robotic affordance understanding are investigated throughout the thesis. The key contributions are summarized as follows:

- **Multi-grasp Affordance Detection**. To achieve real-time inferencing for physical grasping, a grasp region proposal network for identification of potential grasp regions is introduced. The network then partitions the grasp configuration estimation problem into regression over the bounding box parameters, and classification of the orientation angles. The whole network design enables real-time grasp affordance detection for real-world robotic grasping.

- **Affordance Learning via Synthetic data**. Beyond grasp affordance detection, general affordance prediction is considered to account for more commonly encountered robotic manipulation tasks The prediction is formulated as segmentation problem. Due to labor-intensive annotation for pixel-level ground truth, an segmentation architecture is introduced to enable adapting annotations from synthetic data unsupervisedly, while achieving comparable performance to supervisedly learned approaches.

- **Learning toward Multi-affordance and Ranking**. Given multiple affordances may exist on the same object part, the affordance prediction is extended to multiple affordance prediction and ranking, on a single object part. Multiple affordance prediction benefits consecutive planning for realistic scenarios where desired tools are absent. Moreover, the designed architecture generalizes learned affordance to unseen categories, improving the applicability

of learned model.

- **Improving Affordance Detection on Novel Objects**. To generalize the affordance model on unseen categories, the performance loss is observed. To tackle the issue, two modules are introduced to improve the performance: branch-wise attention module and attribute-like auxiliary module. To bridge vision detection and physical robotic manipulation, a system incorporating proposed affordance detector, a pre-trained object detector, and PDDL is introduced. Planning and execution of a sequence of action primitives defined by predicted affordance is enabled.

- **Application: Assistive Manipulation in Human-in-the-loop System**. This chapter explores an application of a previously described component, grasp affordance detection. The whole system design is a case study of a human-in-the-loop system, incorporating the Augmented Reality (AR) glasses, the Tongue-Drive System (TDS) and robotic manipulator with proposed grasp detection. The overall system design is a preliminary study for people with paralysis in upper limb. The chapter details the design and experiments with healthy human subjects.

CHAPTER 8
FUTURE WORK

For the future work of robotic affordance learning, there are several research directions.

Affordance learning in this thesis focuses on identifying functional parts of objects. To properly interact with an object, knowing how to manipulate is also essential and sometimes requires more information. For example, for affordances such as *grasp*, *contain* or *support*, it is straightforward for robot agents to interact with identified object parts and corresponding affordances. However, affordances such as *pound* and *cut* may require additional information of operational directions in order to use the tools properly. This information is not directly provided by the affordance detection. A combination of affordance detection with recent studies of keypoint estimation could serve as a solution. By annotating the direction of operation as with a set of keypoints associated with specific categories or affordances, estimating the operational direction as a vector is enabled. Furthermore, keypoint estimation could be treated as simplified pose estimation. Knowing the pose of object for manipulation benefits achieving complicated manipulation tasks. Meanwhile, pose estimation potentially contributes to closing the loop while reaching the object.

A second interesting direction is to extend to the first-order affordance learning. The first-order affordance learning refers to the affordances learned via direct interactions with an object. This is commonly done with reinforcement learning approaches in simulation or in real-world. Usually there will be no pre-defined action opportunities on object parts and the agent learns from scratch. However, as humans, we discover several ways to interact with surrounding objects by starting with basic knowledge of functionalities of tools. With an effective zero-order affordance detection model, it would be interesting to study how it affects the learning of the first-order model to develop new ways to operate objects in surroundings.

Lastly, in this thesis we show a bottom-up perspective for reasoning action sequences for manipulations. Specifically, we break down the object parts and identify action opportunities of each

part. Given a desired goal, our approach forms a sequence of action primitives for robot execution. Another research direction is a top-down perspective to infer possible goals/manipulations directly from a scene. The outcome is potentially useful for assistive robots to predict or summarize human intent sharing the similar view. To properly reason about possible goals/intents given a scene, discovering object-action pairs is an essential step for further analysis. A possible solution may be a combination of affordance understanding with scene analysis such VQA in computer vision areas.

REFERENCES

[1] K. B. Shimoga, "Robot grasp synthesis algorithms: a survey," *The International Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996.

[2] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *Proceedings of IEEE ICRA*, 2000, pp. 348–353.

[3] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.

[4] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesisa survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[5] S. Caldera, A. Rassau, and D. Chai, "Review of deep learning methods in robotic grasp detection," *Multimodal Technologies and Interaction*, vol. 2, no. 3, p. 57, 2018.

[6] J. Kober and J. Peters, "Imitation and reinforcement learning," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 55–62, 2010.

[7] Z. Ju, C. Yang, Z. Li, L. Cheng, and H. Ma, "Teleoperation of humanoid baxter robot using haptic feedback," in *2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*, 2014, pp. 1–6.

[8] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic grasping of novel objects," in *Advances in NIPS*, 2006, pp. 1209–1216.

[9] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[10] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from RGBD images: learning using a new rectangle representation," in *Proceedings of IEEE ICRA*, 2011, pp. 3304–3311.

[11] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[12] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *Proceedings of IEEE ICRA*, 2015, pp. 1316–1322.

[13] I. Kamon, T. Flash, and S. Edelman, "Learning to grasp using visual information," in *Proceedings of IEEE ICRA*, vol. 3, 1996, pp. 2470–2476.

[14] U. Asif, M. Bennamoun, and F. A. Sohel, "RGB-D object recognition and grasp detection using hierarchical cascaded forests," *IEEE Transactions on Robotics*, 2017.

[15] S. Ekvall and D. Kragic, "Learning and evaluation of the approach vector for automatic grasp generation and planning," in *Proceedings of IEEE ICRA*, 2007, pp. 4715–4720.

[16] K. Huebner and D. Kragic, "Selection of robot pre-grasps using box-based shape approximation," in *IEEE/RSJ IROS*, 2008, pp. 1765–1770.

[17] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, "Learning to grasp objects with multiple contact points," in *Proceedings of IEEE ICRA*, 2010, pp. 5062–5069.

[18] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.

[19] A. t. Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *arXiv preprint arXiv:1706.09911*, 2017.

[20] D. Rao, Q. V. Le, T. Phoka, M. Quigley, A. Sudsang, and A. Y. Ng, "Grasping novel objects with depth segmentation," in *IEEE/RSJ IROS*, 2010, pp. 2578–2585.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in NIPS*, 2012, pp. 1097–1105.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[23] Z. Wang, Z. Li, B. Wang, and H. Liu, "Robot grasp detection using multimodal deep convolutional neural networks," *Advances in Mechanical Engineering*, vol. 8, no. 9, p. 1, 2016.

[24] J. Wei, H. Liu, G. Yan, and F. Sun, "Robotic grasping recognition using multi-modal deep extreme learning machine," *Multidimensional Systems and Signal Processing*, vol. 28, no. 3, pp. 817–833, 2017.

[25] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *IEEE/RSJ IROS*, 2017.

[26] J. Watson, J. Hughes, and F. Iida, "Real-world, real-time robotic grasping with convolutional neural networks," in *Conference Towards Autonomous Robotic Systems*, 2017, pp. 617–626.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of theIEEE conference on CVPR*, 2016, pp. 770–778.

[28] L. Pinto and A. Gupta, "Supersizing self-supervision: learning to grasp from 50k tries and 700 robot hours," in *Proceedings of IEEE ICRA*, 2016, pp. 3406–3413.

[29] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[30] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *IEEE/RSJ IROS*, 2016, pp. 4461–4468.

[31] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: a real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.

[32] U. Viereck, A. t. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images," *arXiv preprint arXiv:1706.04652*, 2017.

[33] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *International Symposium on Robotics Research*, 2017.

[34] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Qt-opt: scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[35] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with large-scale data collection," in *International Symposium on Experimental Robotics*, 2016, pp. 173–184.

[36] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," *arXiv preprint arXiv:1610.04286*, 2016.

[37] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *Proceedings of IEEE ICRA*, 2018, pp. 4243–4250.

[38] K. Fang, Y. Bai, S. Hinterstoisser, S. Savarese, and M. Kalakrishnan, "Multi-task domain adaptation for deep learning of instance grasping from simulation," in *Proceedings of IEEE ICRA*, 2018.

[39] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, *et al.*, "Domain randomization and generative models for robotic grasping," in *IEEE/RSJ IROS*, 2018, pp. 3482–3489.

[40] D. Guo, F. Sun, H. Liu, T. Kong, B. Fang, and N. Xi, "A hybrid deep architecture for robotic grasp detection," in *Proceedings of IEEE ICRA*, 2017, pp. 1609–1614.

[41] J. J. Gibson, "The theory of affordances," *Hilldale, USA*, vol. 1, p. 2, 1977.

[42] E. Ugur and J. Piater, "Bottom-up learning of object categories, action effects and logical rules: from continuous manipulative exploration to symbolic planning," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 2627–2633.

[43] A. Dehban, L. Jamone, A. R. Kampff, and J. Santos-Victor, "Denoising auto-encoders for learning of objects and tools affordances in continuous space," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 4866–4871.

[44] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Preparatory object re-orientation for task-oriented grasping," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, IEEE, 2016, pp. 893–899.

[45] P. Zech, S. Haller, S. R. Lakani, B. Ridge, E. Ugur, and J. Piater, "Computational models of affordance in robotics: a taxonomy and systematic classification," *Adaptive Behavior*, vol. 25, no. 5, pp. 235–271, 2017.

[46] V. Ferrari and A. Zisserman, "Learning visual attributes," in *Advances in NIPS*, 2008, pp. 433–440.

[47] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features.," in *ICRA*, 2015, pp. 1374–1381.

[48] T. Hermans, F. Li, J. M. Rehg, and A. F. Bobick, "Learning contact locations for pushing and orienting unknown objects," in *IEEE-RAS international conference on humanoid robots (humanoids)*, 2013, pp. 435–442.

[49] A. Aldoma, F. Tombari, and M. Vincze, "Supervised learning of hidden and non-hidden 0-order affordances and detection in real scenes," in *Proceedings of IEEE ICRA*, 2012, pp. 1732–1739.

[50] S. R. Lakani, A. J. Rodríguez-Sánchez, and J. Piater, "Can affordances guide object decomposition into semantically meaningful parts?" In *IEEE Winter Conference on Applications of Computer Vision*, 2017, pp. 82–90.

[51] S. Rezapour Lakani, A. J. Rodríguez-Sánchez, and J. Piater, "Towards affordance detection for robot manipulation using affordance for parts and parts for affordance," *Autonomous Robots*, 2018.

[52] S. R. Lakani, A. J. Rodríguez-Sánchez, and J. Piater, "Exercising affordances of objects: a part-based approach," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3465–3472, 2018.

[53] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: a deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on PAMI*, vol. 39, no. 12, pp. 2481–2495, 2017.

[54] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, IEEE, 2016, pp. 2765–2770.

[55] K. Chaudhary, K. Okada, M. Inaba, and X. Chen, "Predicting part affordances of objects using two-stream fully convolutional network with multimodal inputs," in *IEEE/RSJ IROS*, 2018, pp. 3096–3101.

[56] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on PAMI*, vol. 40, no. 4, pp. 834–848, 2018.

[57] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE ICCV*, 2015, pp. 1529–1537.

[58] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Object-based affordances detection with convolutional neural networks and dense conditional random fields," in *International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[59] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Preceedings of the IEEE ICCV*, 2017, pp. 2980–2988.

[60] T.-T. Do, A. Nguyen, I. Reid, D. G. Caldwell, and N. G. Tsagarakis, "Affordancenet: an end-to-end deep learning approach for object affordance detection," *arXiv preprint arXiv:1709.07326*, 2017.

[61] J. Sawatzky, A. Srikantha, and J. Gall, "Weakly supervised affordance detection," in *Proceedings of the IEEE Conference on CVPR*, 2017, pp. 5197–5206.

[62] J. Sawatzky and J. Gall, "Adaptive binarization for weakly supervised affordance segmentation," *arXiv preprint arXiv:1707.02850*, 2017.

[63] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster r-cnn for object detection in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3339–3348.

[64] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[65] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conference on Computer Vision*, Springer, 2012, pp. 611–625.

[66] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. Van der Smagt, D. Cremers, and T. Brox, "Flownet: learning optical flow with convolutional networks," *arXiv preprint arXiv:1504.06852*, 2015.

[67] U. J. Klemming, *Gazebo*, US Patent 6,745,521, 2004.

[68] W. Qiu and A. Yuille, "Unrealcv: connecting computer vision to unreal engine," in *European Conference on Computer Vision*, Springer, 2016, pp. 909–916.

[69] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," *GitHub repository*, 2016.

[70] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[71] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proceedings of IEEE ICRA*, 2017, pp. 3357–3364.

[72] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: visual question answering in interactive environments," in *Proceedings of the IEEE Conference on CVPR*, 2018, pp. 4089–4098.

[73] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," *arXiv preprint arXiv:1605.06457*, 2016.

[74] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: a large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on CVPR*, 2016, pp. 3234–3243.

[75] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: ground truth from computer games," in *European Conference on Computer Vision*, Springer, 2016, pp. 102–118.

[76] A. Handa, V. Pătrăucean, S. Stent, and R. Cipolla, "Scenenet: an annotated model generator for indoor scene understanding," in *Proceedings of IEEE ICRA*, 2016, pp. 5737–5743.

[77] L. Duan, I. W. Tsang, and D. Xu, "Domain transfer multiple kernel learning," *IEEE Transactions on PAMI*, vol. 34, no. 3, pp. 465–479, 2012.

[78] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on CVPR*, 2012, pp. 2066–2073.

[79] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: domain adaptation using asymmetric kernel transforms," in *Proceedings of the IEEE Conference on CVPR*, 2011, pp. 1785–1792.

[80] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: an unsupervised approach," in *Proceedings of the IEEE ICCV*, 2011, pp. 999–1006.

[81] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Proceedings of the IEEE ICCV*, 2013, pp. 2960–2967.

[82] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[83] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," *arXiv preprint arXiv:1409.7495*, 2014.

[84] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[85] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," *arXiv preprint arXiv:1502.02791*, 2015.

[86] T. Xiao, H. Li, W. Ouyang, and X. Wang, "Learning deep feature representations with domain guided dropout for person re-identification," in *Proceedings of the IEEE Conference on CVPR*, 2016, pp. 1249–1258.

[87] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on CVPR*, vol. 1, 2017, p. 4.

[88] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE Conference on CVPR*, vol. 1, 2017, p. 7.

[89] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "Fcns in the wild: pixel-level adversarial and constraint-based adaptation," *arXiv preprint arXiv:1612.02649*, 2016.

[90] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: cycle-consistent adversarial domain adaptation," *arXiv preprint arXiv:1711.03213*, 2017.

[91] Y.-H. Chen, W.-Y. Chen, Y.-T. Chen, B.-C. Tsai, Y.-C. F. Wang, and M. Sun, "No more discrimination: cross city adaptation of road scene segmenters," in *Preceedings of the IEEE ICCV*, 2017, pp. 2011–2020.

[92] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," *arXiv preprint arXiv:1802.10349*, 2018.

[93] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[94] J. Xu, S. Ramos, D. Vázquez, and A. M. López, "Domain adaptation of deformable part-based models," *IEEE transactions on PAMI*, vol. 36, no. 12, pp. 2367–2380, 2014.

[95] A. Raj, V. P. Namboodiri, and T. Tuytelaars, "Subspace alignment based domain adaptation for rcnn detector," *arXiv preprint arXiv:1507.05578*, 2015.

[96] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in NIPS*, 2015, pp. 91–99.

[97] F. Chu, R. Xu, and P. A. Vela, "Real-world multiobject, multigrasp detection," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3355–3362, 2018.

[98] F.-J. Chu, R. Xu, and P. A. Vela, "Learning affordance segmentation for real-world robotic manipulation via synthetic images," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1140–1147, 2019.

[99] F.-J. Chu, R. Xu, L. Seguin, and P. A. Vela, "Toward affordance detection and ranking on novel objects for real-world robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4070–4077, 2019.

[100] F.-J. Chu, R. Xu, and P. A. Vela, "Detecting robotic affordances on novel objects with regional attention and attributes," *arXiv preprint arXiv:1909.05770*, 2019.

[101] F. Chu, R. Xu, Z. Zhang, P. A. Vela, and M. Ghovanloo, "Hands-free assistive manipulator using augmented reality and tongue drive system," in *2018 IEEE/RSJ IROS*, 2018, pp. 5463–5468.

[102] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Aparicio-Ojea, and K. Goldberg, "Dex-Net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems*, 2017.

[103] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on CVPR*, 2014, pp. 580–587.

[104] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE ICCV*, 2015, pp. 1440–1448.

[105] R. L. Lab, *Cornell grasping dataset*, `http://pr.cs.cornell.edu/grasping/rect_data/data.php`, Accessed: 2017-09-01, 2013.

[106] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.

[107] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[108] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: a large-scale hierarchical image database," in *IEEE conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[109] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: convolutional architecture for fast feature embedding," in *Proceedings of the ACM international conference on Multimedia*, ACM, 2014, pp. 675–678.

[110] R. Margolin, L. Zelnik-Manor, and A. Tal, "How to evaluate foreground maps?" In *Proceedings of the IEEE Conference on CVPR*, 2014, pp. 248–255.

[111] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: bridging the reality gap by domain randomization,"

[112] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL-the planning domain definition language," 1998.

[113] L. Bo, X. Ren, and D. Fox, "Unsupervised feature learning for rgb-d based object recognition," in *Experimental Robotics*, Springer, 2013, pp. 387–402.

[114] Z. Sui, O. C. Jenkins, and K. Desingh, "Axiomatic particle filtering for goal-directed robotic manipulation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2015, pp. 4429–4436.

[115] Z. Sui, L. Xiang, O. C. Jenkins, and K. Desingh, "Goal-directed robot manipulation through axiomatic scene estimation," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 86–104, 2017.

[116] Z. Zeng, Z. Zhou, Z. Sui, and O. C. Jenkins, "Semantic robot programming for goal-directed manipulation in cluttered scenes," in *IEEE International Conference on Robotics and Automation*, IEEE, 2018, pp. 7462–7469.

[117] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: an architecture for general intelligence," *Artificial intelligence*, vol. 33, no. 1, pp. 1–64, 1987.

[118] R. E. Fikes and N. J. Nilsson, "Strips: a new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[119] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *IEEE conference on Computer Vision and Pattern Recognition*, vol. 1, 2017, p. 4.

[120] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[121] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE conference on Computer Vision and Pattern Recognition*, 2016.

[122] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision*, Springer, 2016, pp. 102–118.

[123] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[124] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881–2890.

[125] H. Ding, X. Jiang, B. Shuai, A. Qun Liu, and G. Wang, "Context contrasted feature and gated multi-scale aggregation for scene segmentation," in *IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 2393–2402.

[126] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[127] Y. Yuan and J. Wang, "Ocnet: object context network for scene parsing," *arXiv preprint arXiv:1809.00916*, 2018.

[128] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," in *IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7151–7160.

[129] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *IEEE conference on Computer Vision and Pattern Recognition*, 2019, pp. 3146–3154.

[130] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *International Conference on Computer Vision*, IEEE, 2009, pp. 365–372.

[131] N. Kumar, A. Berg, P. N. Belhumeur, and S. Nayar, "Describable visual attributes for face verification and image search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 10, pp. 1962–1977, 2011.

[132] K. Duan, D. Parikh, D. Crandall, and K. Grauman, "Discovering localized attributes for fine-grained recognition," in *IEEE conference on Computer Vision and Pattern Recognition*, 2012, pp. 3474–3481.

[133] H.-T. Cheng, F.-T. Sun, M. Griss, P. Davis, J. Li, and D. You, "Nuactiv: recognizing unseen new activities using semantic attribute-based learning," in *Proceeding of International Conference on Mobile Systems, Applications, and Services*, ACM, 2013, pp. 361–374.

[134] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: powering robust clothes recognition and retrieval with rich annotations," in *IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 1096–1104.

[135] T. Do, A. Nguyen, and I. Reid, "AffordanceNet: an end-to-end deep learning approach for object affordance detection," in *IEEE International Conference on Robotics and Automation*, 2018.

[136] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," 2017.

[137] Y. Li, "Deep reinforcement learning: an overview," *arXiv preprint arXiv:1701.07274*, 2017.

[138] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Learning deep representation for face alignment with auxiliary attributes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 5, pp. 918–930, 2015.

[139] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *IEEE conference on Computer Vision and Pattern Recognition*, 2009, pp. 1778–1785.

[140] L.-J. Li, H. Su, Y. Lim, and L. Fei-Fei, "Objects as attributes for scene classification," in *European Conference on Computer Vision*, Springer, 2010, pp. 57–69.

[141] Y. Sun, L. Bo, and D. Fox, "Attribute based object identification," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 2096–2103.

[142] Y. Lin, C. Tang, F.-J. Chu, and P. A. Vela, "Using synthetic data and deep networks to recognize primitive shapes for object grasping," *arXiv preprint arXiv:1909.08508*, 2019.

[143] Y. Matsumoto, Y. Nishida, Y. Motomura, and Y. Okawa, "A concept of needs-oriented design and evaluation of assistive robots based on icf," in *2011 IEEE International Conference on Rehabilitation Robotics*, IEEE, 2011, pp. 1–6.

[144] Christopher and D. R. Foundation, "One degree of separation: paralysis and spinal cord injury in the united states [online]," in *Available: http://www.christopherreeve.org*, 2011.

[145] D Ding, R. Cooper, B. Kaminski, J. Kanaly, A Allegretti, E Chaves, and S Hubbard, "Integrated control and related technology of assistive devices," *Assistive Technology*, vol. 15, no. 2, pp. 89–97, 2003.

[146] B Yousefi, X. Huo, J Kim, E Veledar, and M. Ghovanloo, "Quantitative and comparative assessment of learning in a tongue-operated computer input device-part ii: navigation tasks," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 4, pp. 633–643, 2012.

[147] T Chen, M Ciocarlie, S Cousins, P Grice, K Hawkins, K Hsiao, C Kemp, C King, D Lazewatsky, A Leeper, H Nguyen, A Paepcke, C Pantofaru, W Smart, and L Takayama, "Robots for humanity: using assistive robotics to empower people with disabilities," *IEEE Robotics and Automation Magazine*, vol. 20, no. 1, pp. 30–39, 2013.

[148] P Deegan, R Grupen, A Hanson, E Horrell, S. Ou, E Riseman, S Sen, B Thibodeau, A Williams, and D Xie, "Mobile manipulators for assisted living in residential settings," *Autonomous Robots*, vol. 24, no. 2, pp. 179–192, 2008.

[149] B Graf, M Hans, and R. Schraft, "Care-o-bot II - development of a next generation robotic home assistant," *Autonomous Robots*, vol. 16, no. 2, pp. 193–205, 2004.

[150] S. C. D. Kent C. Saldanha, "A comparison of remote robot teleoperation interfaces for general object manipulation," in *ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 371–379.

[151] L. N. Andreasen Struijk, L. L. Egsgaard, R. Lontis, M. Gaihede, and B. Bentsen, "Wireless intraoral tongue control of an assistive robotic arm for individuals with tetraplegia," *Journal of Neuroengineering and Rehabilitation*, vol. 14, no. 1, p. 110, 2017.

[152] L. Adreasen Struijk and R. Lontis, "Comparison of tongue interface with keyboard for control of an assistive robotic arm," in *IEEE International Conference on Rehabilitation Robotics*, 2017, pp. 925–928.

[153] C Dune, C Leroux, and E Marchand, "Intuitive human interaction with an arm robot for severly handicapped people - a one click approach," in *IEEE International Conference on Rehabilitation Robotics*, 2007, pp. 582–589.

[154] P. Grice and C. Kemp, "Assistive mobile manipulation: designing for operators with motor impairments," in *RSS Workshop on Socially and Physically Assistive Robotics for Humanity*, 2016.

[155] C. Chung, H. Wang, and R. Cooper, "Functional assessment and performance evaluation for assistive robotic manipulators: literature review," *Journal of Spinal Cord Medicine*, vol. 36, no. 4, pp. 273–289, 2013.

[156] C. Martins Pereira, R Bolliger Neto, A. Reynaldo, M. de Miranda Luzo, and R. Oliveira, "Development and evaluation of a head-controlled human-computer interface with mouse-like functions for physically disabled users," *Clinics*, vol. 64, no. 10, pp. 975–981, 2009.

[157] X. Huo, J. Wang, and M. Ghovanloo, "A magneto-inductive sensor based wireless tongue-computer interface," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 5, pp. 497–504, 2008.

[158] X. Huo and M. Ghovanloo, "Evaluation of a wireless wearable tongue–computer interface by individuals with high-level spinal cord injuries," *Journal of Neural Engineering*, vol. 7, no. 2, p. 026 008, 2010.

[159] J. Kim, H. Park, J. Bruce, E. Sutton, D. Rowles, D. Pucci, J. Holbrook, J. Minocha, B. Nardone, D. West, *et al.*, "The tongue enables computer and wheelchair control for people with spinal cord injury," *Science Translational Medicine*, vol. 5, no. 213, 213ra166–213ra166, 2013.

[160] K. Lyons and S. Joshi, "Paralyzed subject controls telepresence mobile robot using novel sEMG brain-computer interface: case study," in *IEEE International Conference on Rehabilitation Robotics*, 2013.

[161] I Laffont, N Biard, G Chalubert, L Delahoche, B Marhic, F. Boyer, and C Leroux, "Evaluation of a graphic interface to control a robotic grasping arm: a multicenter study," *Archives of Physical Medicine and Rehabilitation*, vol. 90, no. 10, pp. 1740–1748, 2009.

[162] M. Markovic, S. Dosen, C. Cipriani, D. Popovic, and D. Farina, "Stereovision and augmented reality for closed-loop control of grasping in hand prostheses," *Journal of Neural Engineering*, vol. 11, no. 4, p. 046 001, 2014.

[163] F Leishman, V Monfort, O Horn, and G Bourhis, "Driving assistance by deictic control for a smart wheelchair: the assessment issue," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 1, pp. 66–77, 2014.

[164] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[165] R Munoz-Salinas, "Aruco: a minimal library for augmented reality applications based on opencv," *Universidad de Crdoba*, 2012.

[166] L. Keselman, "Motion planning for redundant manipulators and other high degree-of-freedom systems," Master's thesis, Georgia Institute of Technology, 2014.

[167] L. Keselman, E. Verriest, and P. Vela, "Forage RRT - an efficient approach to task-space goal planning for high dimensional systems," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 1572–1577.

[168] J. Weisz, P. K. Allen, A. G. Barszap, and S. S. Joshi, "Assistive grasping with an augmented reality user interface," *The International Journal of Robotics Research*, p. 0 278 364 917 707 024, 2017.

[169] Y.-S. L.-K. Cio, M. Raison, C. L. Ménard, and S. Achiche, "Proof of concept of an assistive robotic arm control using artificial stereovision and eye-tracking," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 12, pp. 2344–2352, 2019.

[170] U. Viereck, A. t. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images," in *Conference on Robot Learning*, 2017, pp. 373–385.

[171] Y. Wang, H. Zeng, A. Song, B. Xu, H. Li, L. Zhu, P. Wen, and J. Liu, "Robotic arm control using hybrid brain-machine interface and augmented reality feedback," in *IEEE//EMBS International Conference on Neural Engineering*, 2017, pp. 411–414.